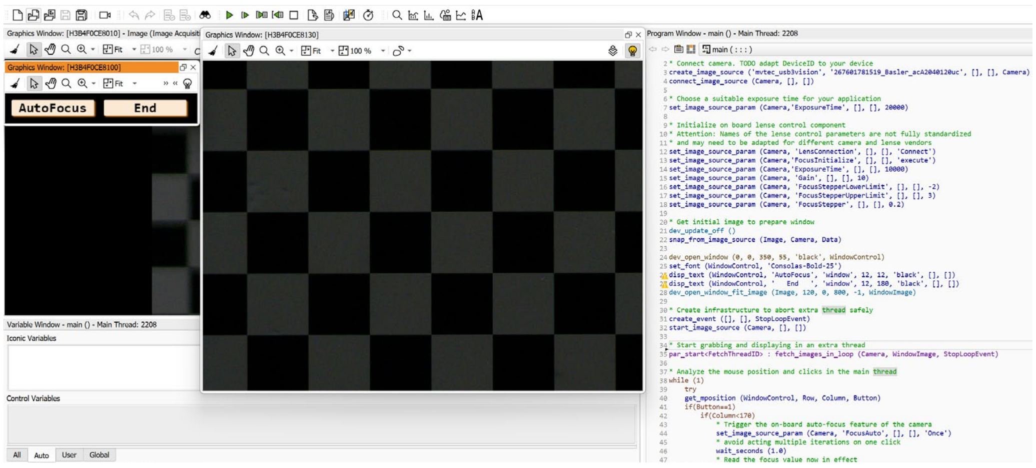
Application Note: Liquid lens control using Halcon and Merlic


Graphics Window: [H3B4F0CE8100]
AutoFocus
End
Graphics Window: [H3B4F0CE8130]
Variable Window - main () - Main Thread: 2208
Iconic Variables
Control Variables
All
Auto
User
Global
Program Window - main () - Main Thread: 2208

```
* Connect camera. TODO adapt DeviceID to your device
create_image_source ('mvtec_usb3vision', '267601781519_Basler_acA2040120uc', [], [], Camera)
connect_image_source (Camera, [], [])

* Choose a suitable exposure time for your application
set_image_source_param (Camera,'ExposureTime', [], [], 20000)

* Initialize on board lense control component
* Attention: Names of the lense control parameters are not fully standardized
* and may need to be adapted for different camera and lense vendors
set_image_source_param (Camera, 'LensConnection', [], [], 'Connect')
set_image_source_param (Camera,'FocusInitialize', [], [], 'execute')
set_image_source_param (Camera,'ExposureTime', [], [], 10000)
set_image_source_param (Camera, 'Gain', [], [], 10)
set_image_source_param (Camera, 'FocusStepperLowerLimit', [], [], -2)
set_image_source_param (Camera, 'FocusStepperUpperLimit', [], [], 3)
set_image_source_param (Camera, 'FocusStepper', [], [], 0.2)

* Get initial image to prepare window
dev_update_off ()
snap_from_image_source (Image, Camera, Data)

dev_open_window (0, 0, 350, 55, 'black', WindowControl)
set_font (WindowControl, 'Consolas-Bold-25')
disp_text (WindowControl, 'AutoFocus', 'window', 12, 12, 'black', [], [])
disp_text (WindowControl, '   End   ', 'window', 12, 180, 'black', [], [])
dev_open_window_fit_image (Image, 120, 0, 800, -1, WindowImage)

* Create infrastructure to abort extra thread safely
create_event ([], [], StopLoopEvent)
start_image_source (Camera, [], [])

* Start grabbing and displaying in an extra thread
par_start<FetchThreadID> : fetch_images_in_loop (Camera, WindowImage, StopLoopEvent)

* Analyze the mouse position and clicks in the main thread
while (1)
    try
    get_mposition (WindowControl, Row, Column, Button)
    if(Button==1)
        if(Column<170)
            * Trigger the on-board auto-focus feature of the camera
            set_image_source_param (Camera, 'FocusAuto', [], [], 'Once')
            * avoid acting multiple iterations on one click
            wait_seconds (1.0)
            * Read the focus value now in effect
```

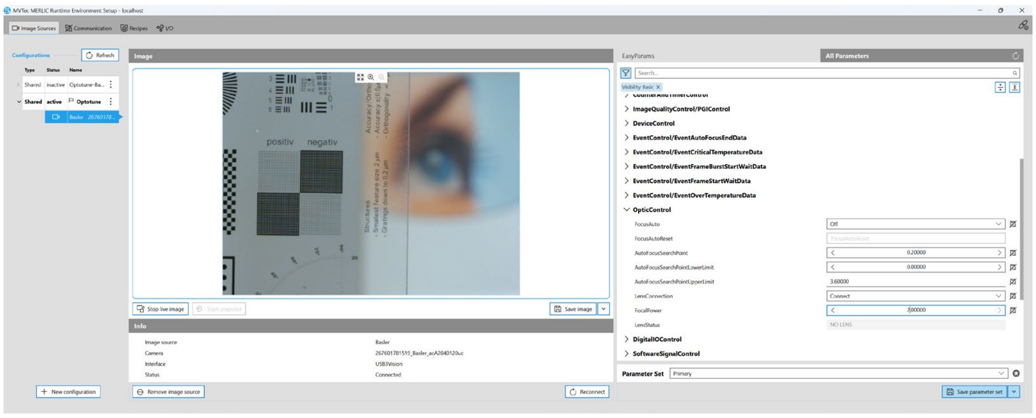


Image source	Basler
Camera	267601781519_Basler_acA2040120uc
Interface	USB3Vision
Status	Connected

ImageQualityControl/PGIControl
DeviceControl
EventControl/EventAutoFocusEndData
EventControl/EventCriticalTemperatureData
EventControl/EventFrameBurstStartWaitData
EventControl/EventFrameStartWaitData
EventControl/EventOverTemperatureData
OpticControl

FocusAuto	Off
FocusAutoReset	FocusAutoReset
AutoFocusSearchPoint	0.20000
AutoFocusSearchPointLowerLimit	0.00000
AutoFocusSearchPointUpperLimit	3.60000
LensConnection	Connect
FocalPower	2.00000
LensStatus	NO LENS

DigitalIOControl
SoftwareSignalControl

Parameter Set: Primary

Table of Contents

1 Introduction 3
1.1 Integration of Optotune Liquid Lenses and Cameras 3
1.2 Optotune-MVTec Interface Options 3
2 Interfacing Optotune Lenses with Halcon 5
2.1 Focal power control and AF through the camera interface 5
2.2 Focal power control through the camera interface and AF using Halcon 7
2.3 Lens control using serial communication of ECC-1C 9
3 Interfacing Optotune Lenses with Merlic 10
4 Conclusion 11

1 Introduction

Machine vision has become an indispensable part of modern automation, enabling systems to “see” and make intelligent decisions without human intervention. From high-speed quality inspection in manufacturing lines to complex 3D object recognition in logistics, machine vision software provides the algorithms and tools needed to translate camera images into actionable data. Among the leaders in this field is MVTec Software GmbH, whose solutions are known for their flexibility, robustness, and broad range of supported applications. Their flagship products—HALCON and MERLIC—address both the demands of expert developers and the needs of users seeking rapid, no-code deployment.

HALCON is MVTec’s comprehensive standard software for machine vision, designed for developers who require maximum flexibility and control. It offers a vast library of operators covering image acquisition, preprocessing, segmentation, feature extraction, 3D vision, and deep learning. HALCON supports a wide range of programming languages and operating systems, making it suitable for integration into diverse industrial environments. For example, HALCON powers applications such as automated defect detection in semiconductor wafers, robot guidance in automotive assembly, and OCR (optical character recognition) for high-speed postal sorting. Its modularity and performance optimization allow solutions to scale from lab prototypes to high-throughput production systems.

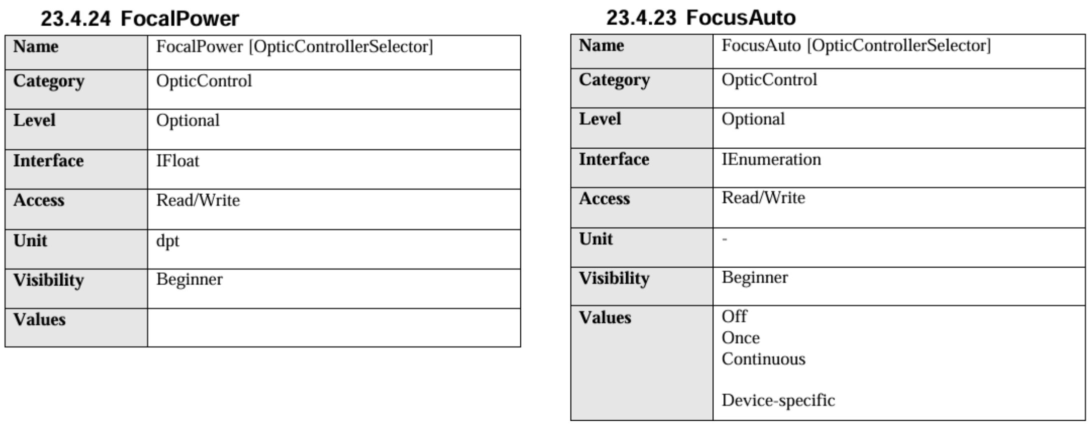
MERLIC, on the other hand, is designed for fast, intuitive machine vision application development without the need for programming. Its drag-and-drop interface, built-in tool library, and easy integration with automation hardware make it ideal for users who want to deploy solutions quickly. MERLIC also incorporates advanced features like deep learning classification, measurement tools, and multi-camera support. Applications include inspecting labels and packaging in the food industry, verifying assembly completeness in electronics manufacturing, and monitoring fill levels in pharmaceutical production. This approach empowers companies to create and adjust vision applications directly on the shop floor, reducing development cycles and lowering the barrier to entry.

Together, HALCON and MERLIC form a complementary suite that not only covers a wide range of machine vision needs but also integrates seamlessly with advanced optical hardware such as **Optotune liquid lenses**. Optotune tunable lenses enable fast, software-controlled focus adjustments, allowing vision systems to adapt to varying object heights or inspection depths without mechanical movement. By combining MVTec’s powerful vision algorithms with the dynamic focusing capabilities of Optotune lenses, users can build inspection systems for tasks like multi-layer PCB inspection, automated bottle fill checks, or 3D part measurements—delivering both speed and precision while simplifying system design.

1.1 Integration of Optotune Liquid Lenses and Cameras

At the heart of modern camera integration lies GenICam (Generic Interface for Cameras), a standard managed by the EMVA. It abstracts hardware interfaces (like GigE Vision, USB3 Vision) from the application layer, offering a unified programming interface through modules like GenApi, SFNC (Standard Feature Naming Convention), and GenTL.
A key benefit for liquid lens integration is that cameras conforming to GenICam can expose optics control parameters—such as focal power or lens focus adjustments—as standard features. This allows software applications, SDKs, or high-level interfaces to control the liquid lens directly through the camera’s GenICam interface, streamlining integration and enabling flexible focus adjustments within existing vision systems.

GenICam Standard Features Naming Convention (SFNC) version 2.7 presents OPTIC CONTROL standard naming convention for controlling liquid lens focal power, autofocus (AF) features, etc. This standard naming convention facilitates integration of Optotune Liquid Lenses and Cameras and especially accessing the features through 3rd-party software such as Halcon or Merlic.



23.4.24 FocalPower

Name	FocalPower [OpticControllerSelector]
Category	OpticControl
Level	Optional
Interface	IFloat
Access	Read/Write
Unit	dpt
Visibility	Beginner
Values	

23.4.23 FocusAuto

Name	FocusAuto [OpticControllerSelector]
Category	OpticControl
Level	Optional
Interface	IEnumeration
Access	Read/Write
Unit	-
Visibility	Beginner
Values	Off Once Continuous Device-specific

Figure 1. Examples of GenICAM SFNC commands for liquid lens control

Optotune has been working with many camera manufacturers to create an ecosystem for controlling liquid lenses through the camera interface. A marketing video on this topic can be found in this link. To name a few successful examples, we can mention Basler, Opto-Engineering ITALA, and Allied Vision Alvium cameras. In the case of Opto-Engineering ITALA G.EL cameras, a current driver is embedded inside the camera and the camera acts as a controller for Optotune liquid lenses. Using the camera interface, a full control of liquid lens current, focal power, and AF features is available. On the other hand, for Basler or Allied Vision, Optotune ECC-1C controller (which is embedded inside the housing of the liquid lens) is required to provide current to the liquid lens. ECC-1C can be connected to the GPIO port of the camera (normally using a Y-cable to provide power), and communication between the camera and ECC-1C can be established using I2C or UART.

By combining Optotune's tunable optics with GenICam-compliant cameras and the powerful vision capabilities of MVTec's HALCON or MERLIC software, engineers can create adaptable, compact, and highly reliable machine vision systems. This synergy enables seamless control of focus, rapid adaptation to varying inspection depths, and efficient integration into existing workflows. From robotics and precision inspection to logistics and automated quality control, these solutions open new possibilities for speed, accuracy, and flexibility in industrial vision.

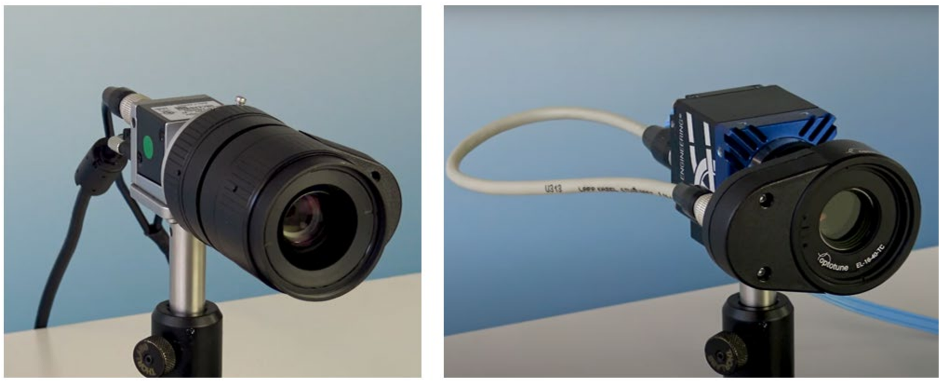


Figure 2. Integration of Optotune Liquid Lenses with Basler (left) and ITALA (right) cameras

1.2 Optotune-MVTec Interface Options

There are three main scenarios for operating Optotune liquid lenses to adjust focal power or implement an autofocus (AF) routine using MVTec's HALCON or MERLIC software.

Scenario 1. Focal power control and AF through the camera interface: The liquid lens is connected directly to a camera that supports Optotune integration and has an AF algorithm built into its firmware. In this setup, the AF routine runs entirely on the camera, and the PC (host) is only used to send the necessary commands for the application—no image processing or algorithm execution occurs on the host. Both HALCON and MERLIC support this configuration.

Scenario 2. Focal power control through the camera interface and AF using Halcon: The liquid lens is connected to the camera, and its focal power can be adjusted via the camera interface, but the AF algorithm runs on the PC using HALCON. This approach requires a camera that provides focal power control over its interface but does not need to have an onboard AF algorithm.

Scenario 3. Lens control using serial communication of ECC-1C: The camera and the ECC-1C lens controller are each connected separately to the PC. The ECC-1C communicates with the host via serial connection using Optotune's UART-USB cable. In this case, the AF algorithm runs on the PC in HALCON, making use of the host's processing power. This method works with any camera, regardless of whether it has built-in Optotune lens integration.

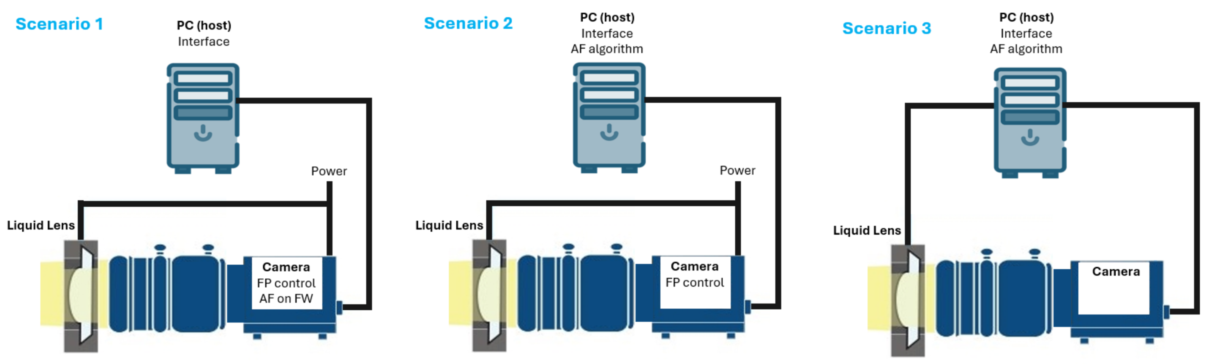


Figure 3. Three different scenarios for using Optotune liquid lenses through Halcon

We provide example Halcon codes for these three scenarios and discuss them in the next section. On the other hand, for Merlic only the first scenario is available, and it will be discussed in section 3.

2 Optotune Lenses with Halcon

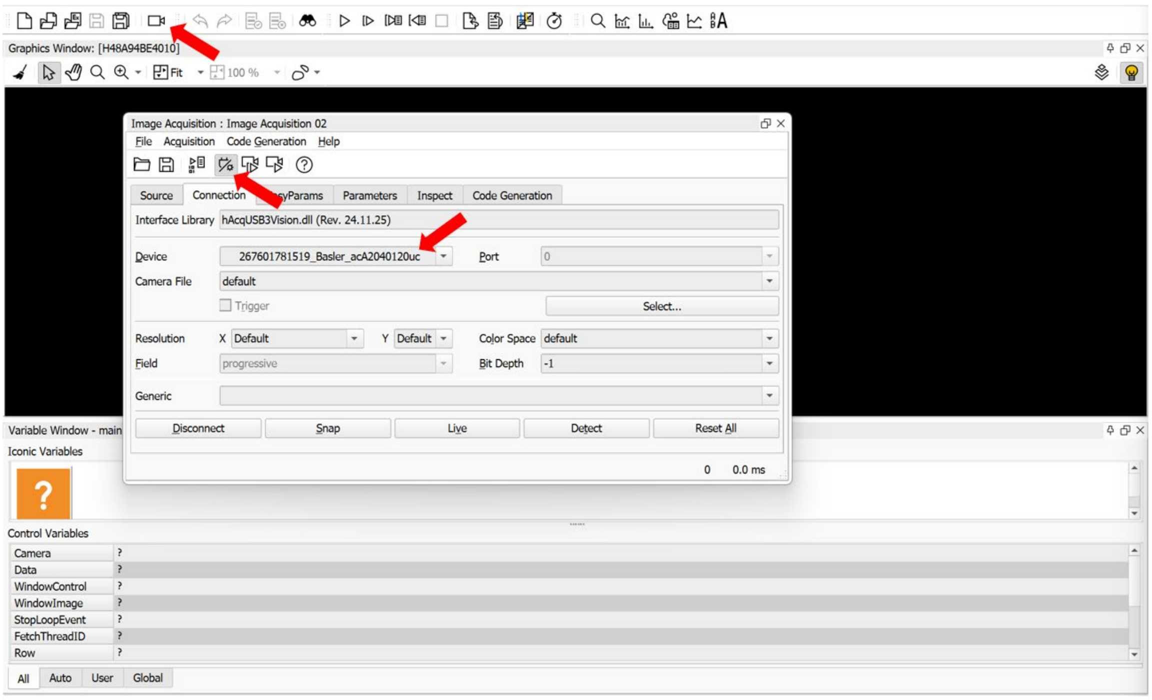
2.1 Focal power control and AF through the camera interface

In this section, we discuss scenario 1 for using Optotune liquid lenses with Halcon. HDevelop XL 25.05 Progress is used throughout these examples. As discussed in section 1.2, scenario 1 is only available for cameras which have the possibility of integration with Optotune liquid lenses (through ECC-1C controller or with a built-in lens controller) to control the focal power of the liquid lens and also have an AF algorithm implemented on their firmware.

Example script for this scenario can be found as file **AutoFocus_Camera.hdev** here: Link

We dive deep into this script and discuss different commands.

First, ensure that the camera connection is verified in the **Image Acquisition** tab. This step also allows you to confirm the **DeviceID**, which can then be used in your script.



Interface Library	hAcqUSB3Vision.dll (Rev. 24.11.25)
Device	267601781519_Basler_acA2040120uc
Port	0
Camera File	default
Trigger	☐
Resolution X	Default
Resolution Y	Default
Color Space	default
Field	progressive
Bit Depth	-1
Generic	

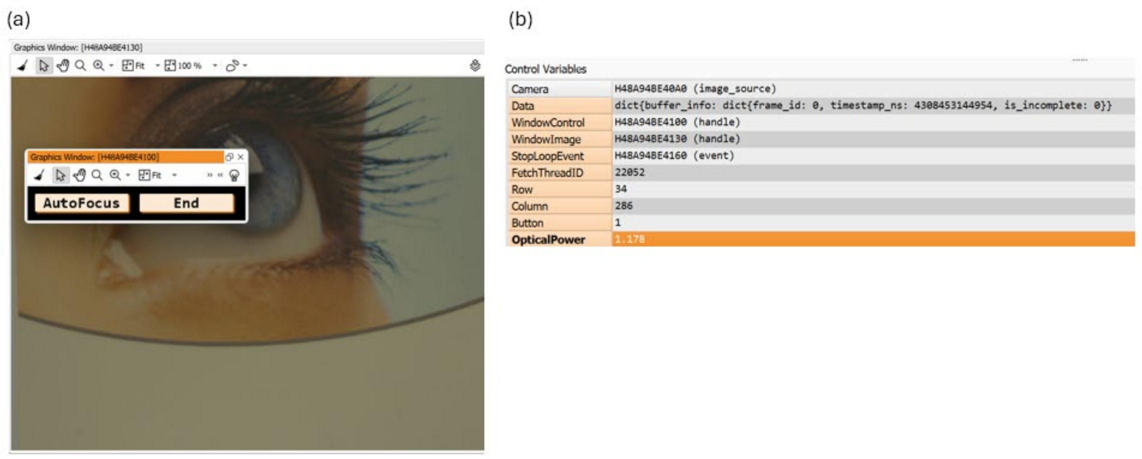
0 0.0 ms

Camera	?
Data	?
WindowControl	?
WindowImage	?
StopLoopEvent	?
FetchThreadID	?
Row	?

Figure 4. Image Acquisition and camera detection

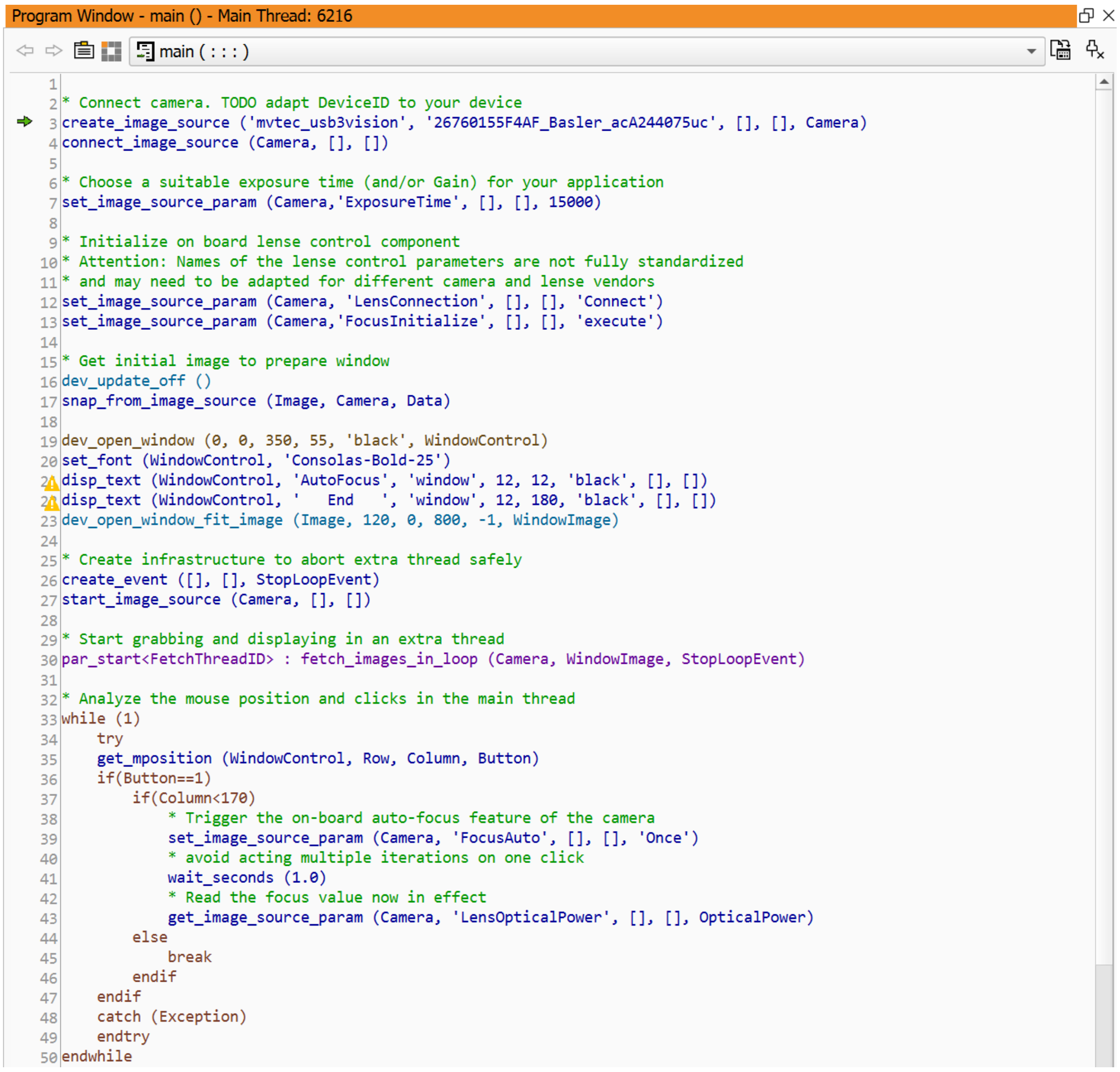
The camera should be connected using the DeviceID with *create_image_source* and also *connect_image_source*. After this step, it is important that depending on the light conditions, Exposure time and/or gain of the camera is selected. As an example, in the script below, Exposure time of 15ms is selected. All the camera parameters, such as exposure time, gain, whitebalance, etc, and also camera parameters for liquid lens control can be adjusted by *set_image_source_param* commands. The main command which needs to be used for Autofocus is *set_image_source_param(Camera, 'FocusAuto', [], [], 'Once')*.

A while loop is used to create a control window with Autofocus button and the best focal power after AF can be found by reading the focal power value using *get_image_source_param(Camera, 'LensOpticalPower', [], [], OpticalPower)* command. This helps to represent the best focus focal power as a control parameter, Optical power' as shown in Figure 5. (b).


(a)
(b)
Graphics Window: [H48A94BE4130]
Graphics Window: [H48A94BE4100]
AutoFocus
End
Control Variables

Camera	H48A94BE40A0 (image_source)
Data	dict{buffer_info: dict{frame_id: 0, timestamp_ns: 4308453144954, is_incomplete: 0}}
WindowControl	H48A94BE4100 (handle)
WindowImage	H48A94BE4130 (handle)
StopLoopEvent	H48A94BE4160 (event)
FetchThreadID	22052
Row	34
Column	286
Button	1
OpticalPower	1.178

Figure 5. (a) Image representation and AF. (b) Control variables



```
* Connect camera. TODO adapt DeviceID to your device
create_image_source ('mvtec_usb3vision', '26760155F4AF_Basler_acA244075uc', [], [], Camera)
connect_image_source (Camera, [], [])

* Choose a suitable exposure time (and/or Gain) for your application
set_image_source_param (Camera,'ExposureTime', [], [], 15000)

* Initialize on board lense control component
* Attention: Names of the lense control parameters are not fully standardized
* and may need to be adapted for different camera and lense vendors
set_image_source_param (Camera, 'LensConnection', [], [], 'Connect')
set_image_source_param (Camera,'FocusInitialize', [], [], 'execute')

* Get initial image to prepare window
dev_update_off ()
snap_from_image_source (Image, Camera, Data)

dev_open_window (0, 0, 350, 55, 'black', WindowControl)
set_font (WindowControl, 'Consolas-Bold-25')
disp_text (WindowControl, 'AutoFocus', 'window', 12, 12, 'black', [], [])
disp_text (WindowControl, '   End   ', 'window', 12, 180, 'black', [], [])
dev_open_window_fit_image (Image, 120, 0, 800, -1, WindowImage)

* Create infrastructure to abort extra thread safely
create_event ([], [], StopLoopEvent)
start_image_source (Camera, [], [])

* Start grabbing and displaying in an extra thread
par_start<FetchThreadID> : fetch_images_in_loop (Camera, WindowImage, StopLoopEvent)

* Analyze the mouse position and clicks in the main thread
while (1)
    try
    get_mposition (WindowControl, Row, Column, Button)
    if(Button==1)
        if(Column<170)
            * Trigger the on-board auto-focus feature of the camera
            set_image_source_param (Camera, 'FocusAuto', [], [], 'Once')
            * avoid acting multiple iterations on one click
            wait_seconds (1.0)
            * Read the focus value now in effect
            get_image_source_param (Camera, 'LensOpticalPower', [], [], OpticalPower)
        else
            break
        endif
    endif
    catch (Exception)
    endtry
endwhile
```

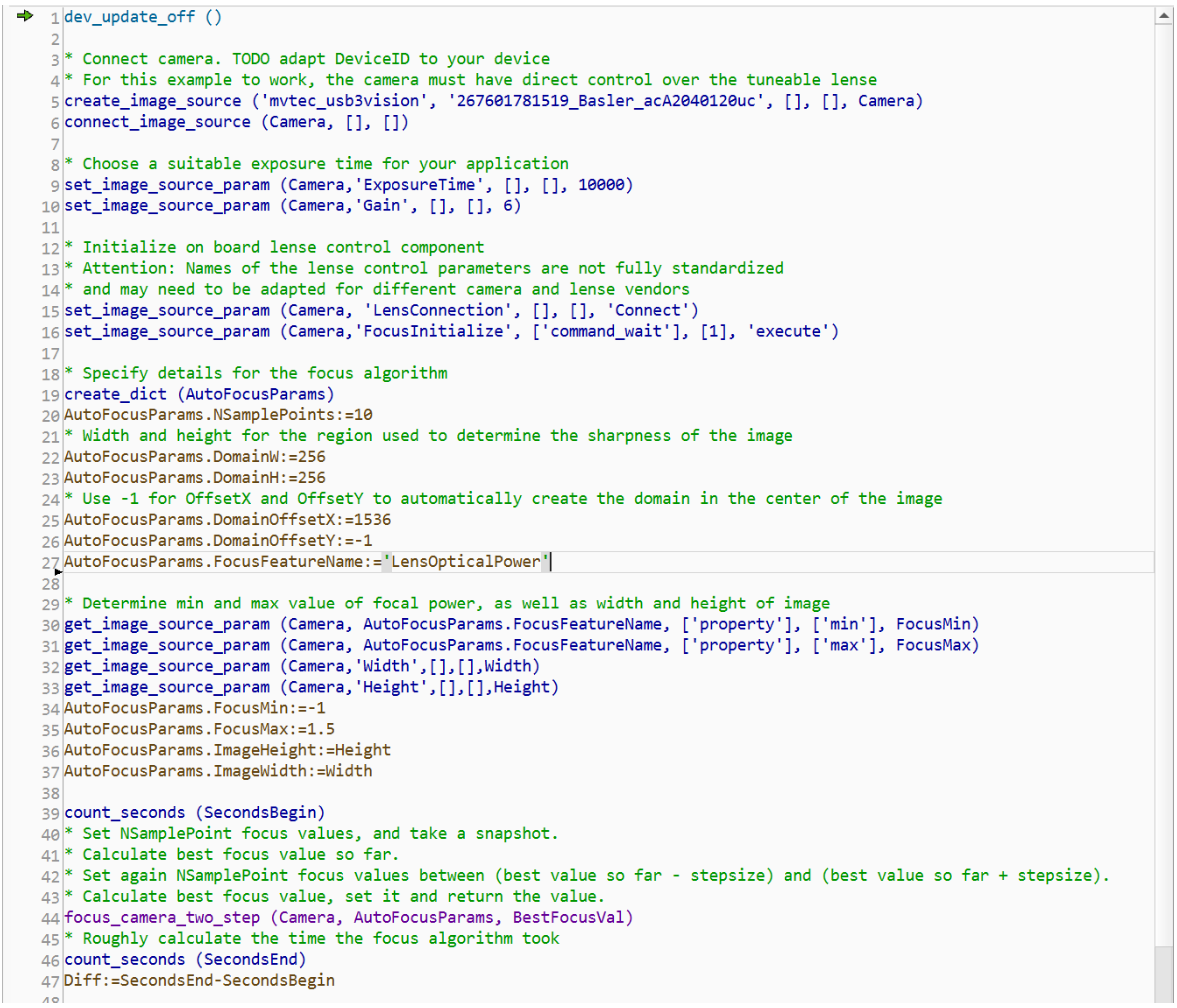
Figure 6. Hdevelop scripts for focal power control and AF through the camera interface

2.2 Focal power control through the camera interface and AF using Halcon

In this section, we discuss scenario 2 for using Optotune liquid lenses with Halcon. HDevelop XL 25.05 Progress is used throughout these examples. As discussed in section 1.2, scenario 2 is available for cameras which have the possibility of integration with Optotune liquid lenses (through ECC-1C controller or with a built-in lens controller) to control the focal power of the liquid lens. However, the AF algorithm is implemented with a Halcon algorithm.

Example script for this scenario can be found as file **AutoFocus_Camera_Manual.hdev** here: Link

We dive deep into this script and discuss different commands. Let's start with an overview of the AF algorithm. The example autofocus (AF) algorithm presented here uses a two-step process to identify the optimal focus, or focal power value, within a user-defined range. The user specifies the minimum and maximum focal power values, as well as the step size. In the first step, a coarse sweep, the algorithm increments the focal power across the specified range and calculates a Sobel-filter–based contrast metric at each point. A Lorentzian curve is then fitted to the resulting contrast values to estimate the focal power corresponding to maximum contrast. In the second step, a fine sweep, the algorithm performs a higher-resolution search around this estimated focal power to pinpoint the exact value that yields the maximum contrast.



```
dev_update_off ()

* Connect camera. TODO adapt DeviceID to your device
* For this example to work, the camera must have direct control over the tuneable lense
create_image_source ('mvtec_usb3vision', '267601781519_Basler_acA2040120uc', [], [], Camera)
connect_image_source (Camera, [], [])

* Choose a suitable exposure time for your application
set_image_source_param (Camera,'ExposureTime', [], [], 10000)
set_image_source_param (Camera,'Gain', [], [], 6)

* Initialize on board lense control component
* Attention: Names of the lense control parameters are not fully standardized
* and may need to be adapted for different camera and lense vendors
set_image_source_param (Camera, 'LensConnection', [], [], 'Connect')
set_image_source_param (Camera,'FocusInitialize', ['command_wait'], [1], 'execute')

* Specify details for the focus algorithm
create_dict (AutoFocusParams)
AutoFocusParams.NSamplePoints:=10
* Width and height for the region used to determine the sharpness of the image
AutoFocusParams.DomainW:=256
AutoFocusParams.DomainH:=256
* Use -1 for OffsetX and OffsetY to automatically create the domain in the center of the image
AutoFocusParams.DomainOffsetX:=1536
AutoFocusParams.DomainOffsetY:=-1
AutoFocusParams.FocusFeatureName:='LensOpticalPower'

* Determine min and max value of focal power, as well as width and height of image
get_image_source_param (Camera, AutoFocusParams.FocusFeatureName, ['property'], ['min'], FocusMin)
get_image_source_param (Camera, AutoFocusParams.FocusFeatureName, ['property'], ['max'], FocusMax)
get_image_source_param (Camera,'Width',[],[],Width)
get_image_source_param (Camera,'Height',[],[],Height)
AutoFocusParams.FocusMin:=-1
AutoFocusParams.FocusMax:=1.5
AutoFocusParams.ImageHeight:=Height
AutoFocusParams.ImageWidth:=Width

count_seconds (SecondsBegin)
* Set NSamplePoint focus values, and take a snapshot.
* Calculate best focus value so far.
* Set again NSamplePoint focus values between (best value so far - stepsize) and (best value so far + stepsize).
* Calculate best focus value, set it and return the value.
focus_camera_two_step (Camera, AutoFocusParams, BestFocusVal)
* Roughly calculate the time the focus algorithm took
count_seconds (SecondsEnd)
Diff:=SecondsEnd-SecondsBegin
```

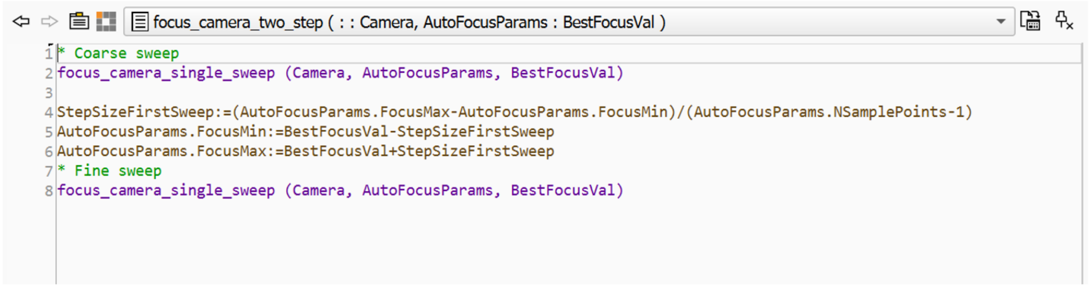
Figure 7. Hdevelop scripts for focal power control through the camera interface and AF using Halcon

The autofocus (AF) operation is performed using the *focus_camera_two_step* function. To achieve optimal results, it is essential to configure several key parameters. First, define the minimum and maximum values of the AF range using *AutoFocusParams.FocusMin* and *AutoFocusParams.FocusMax*. Next, specify the size and position of the autofocus region of interest (ROI) by adjusting *AutoFocusParams.DomainW*, *AutoFocusParams.DomainH*, *AutoFocusParams.DomainOffsetX*, and *AutoFocusParams.DomainOffsetY*. If the offset values are set to *-1*, the system will automatically position the AF ROI at the center of the image, ensuring a convenient default setup.

Additionally, make sure that the *FocusFeatureName* parameter is correctly configured, as this can differ between camera models. According to the GeniCAM standard (as noted in Section 1.1), the expected parameter name is *FocalPower*. However, as demonstrated in the example above, some cameras—such as Basler models—use an alternative name, like *LensOpticalPower*. Verifying this setting for your specific device is crucial for successful autofocus functionality.

In the back-end of *focus_camera_two_step* function, as indicated in Figure 8, there are two steps of a coarse and a fine sweep. These two steps are basically similar except for the fact that the Step size and the Min/Max range for them are different. In each of these steps, a Sobel-filter based contrast metric is evaluated to find the best focus using a for-loop. Sobel filter is available in Halcon.

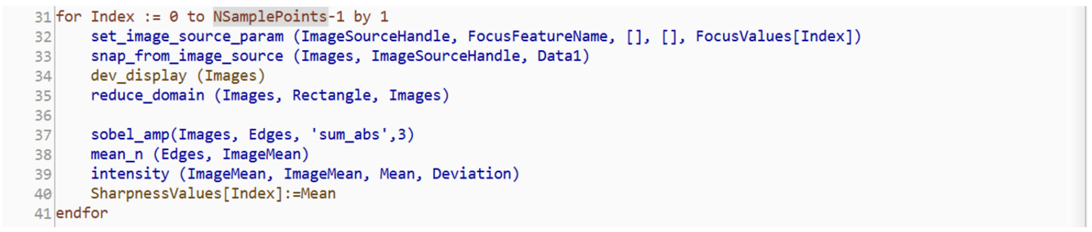
(a)


focus_camera_two_step (: : Camera, AutoFocusParams : BestFocusVal)

```
* Coarse sweep
focus_camera_single_sweep (Camera, AutoFocusParams, BestFocusVal)

StepSizeFirstSweep:=(AutoFocusParams.FocusMax-AutoFocusParams.FocusMin)/(AutoFocusParams.NSamplePoints-1)
AutoFocusParams.FocusMin:=BestFocusVal-StepSizeFirstSweep
AutoFocusParams.FocusMax:=BestFocusVal+StepSizeFirstSweep
* Fine sweep
focus_camera_single_sweep (Camera, AutoFocusParams, BestFocusVal)
```

(b)



```
for Index := 0 to NSamplePoints-1 by 1
    set_image_source_param (ImageSourceHandle, FocusFeatureName, [], [], FocusValues[Index])
    snap_from_image_source (Images, ImageSourceHandle, Data1)
    dev_display (Images)
    reduce_domain (Images, Rectangle, Images)

    sobel_amp(Images, Edges, 'sum_abs',3)
    mean_n (Edges, ImageMean)
    intensity (ImageMean, ImageMean, Mean, Deviation)
    SharpnessValues[Index]:=Mean
endfor
```

Figure 8. Two-step AF algorithm based on the Sobel-filter

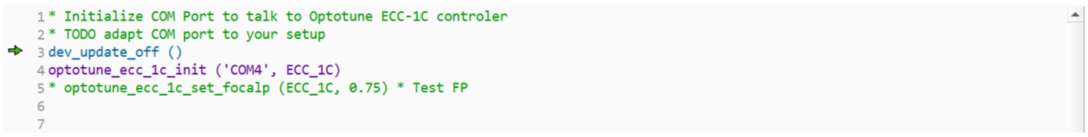
2.3 Lens control using serial communication of ECC-1C

In this section, we discuss scenario 3 for using Optotune liquid lenses with Halcon. HDevelop XL 25.05 Progress is used throughout these examples. As discussed in section 1.2, scenario 3 is available for any camera and the liquid lens control is done with serial communication of ECC-1C. The AF algorithm is implemented with a Halcon algorithm running on the PC (host). The serial communication of ECC-1C can be found in its Datasheet.

Example script for this scenario can be found in **AutoFocus_OptotuneECC_1C_Serial.hdev** here: Link

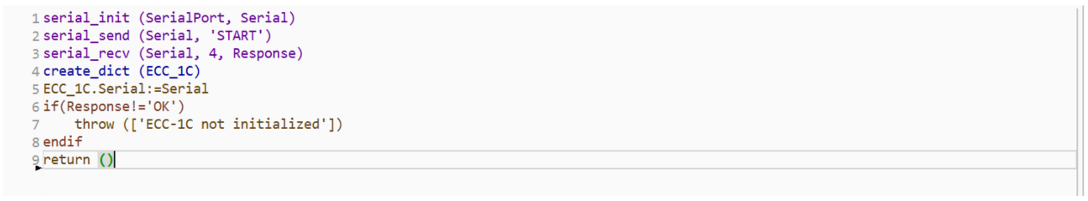
The AF algorithm is exactly like the one which is discussed in section 2.2 as a two-step process for finding the optimal focus using a coarse and a find sweep. Such an algorithm with Optotune ECC-1C is presented in a python example here. The main difference here with what was described in section 2.2 is that the liquid lens focal power control is through serial communication with ECC-1C. It is important that the COM port is selected correctly in the corresponding command, *optotune_ecc_1c_init ('COM4', ECC_1C)*. Other parts such as the camera properties and also *AutoFocusParams* are exactly the same as what was discussed in section 2.1 and section 2.2. Additionally, to set the focal power to a specific value, a command such as *optotune_ecc_1c_set_focalp (ECC_1C, 0.75)* should be used.

(a)



```
* Initialize COM Port to talk to Optotune ECC-1C controler
* TODO adapt COM port to your setup
dev_update_off ()
optotune_ecc_1c_init ('COM4', ECC_1C)
* optotune_ecc_1c_set_focalp (ECC_1C, 0.75) * Test FP
```

(b)



```
serial_init (SerialPort, Serial)
serial_send (Serial, 'START')
serial_recv (Serial, 4, Response)
create_dict (ECC_1C)
ECC_1C.Serial:=Serial
if(Response!='OK')
    throw (['ECC-1C not initialized'])
endif
return ()
```

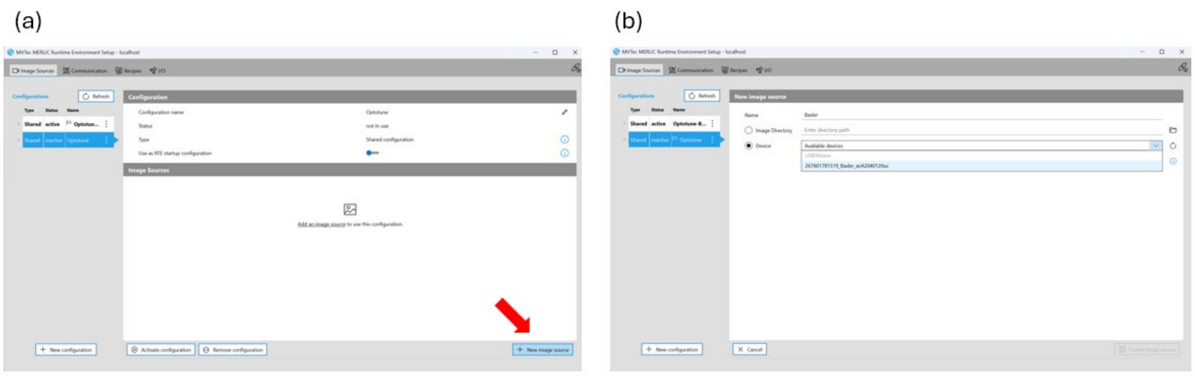
Figure 9. Serial connection with ECC-1C

3 Optotune Lenses with MERLIC

MERLIC is MVTec's intuitive and all-in-one machine vision software designed for users who prefer a graphical, no-code environment. It allows developers and operators to build powerful vision applications without the need for advanced programming skills. With its drag-and-drop interface and integrated tools, MERLIC supports essential machine vision tasks such as inspection, measurement, and positioning. When combined with Optotune liquid lenses, MERLIC can control focus dynamically through GenICam-compliant cameras, enabling rapid adaptation to varying object distances and improving efficiency in applications like quality inspection, logistics, and automated assembly.

Optotune liquid lenses can be controlled in MERLIC only through the camera interface, as described in Scenarios 1 and 2. In this setup, the ECC-1C must be connected to the camera, and all communication with the lens occurs via the camera interface. MERLIC provides access to camera features, including liquid lens parameters. If the camera firmware includes an autofocus (AF) algorithm, MERLIC can utilize it; otherwise, only manual focal power adjustment is available.

In order to start using Optotune liquid lens in MERLIC, firstly, a new image source should be added in the Runtime Environment Setup and the camera with liquid lens control should be selected among the available devices, as shown in Figure 10. After that, the configuration should be activated. This can be done by pressing Activate the configuration to connect this device and configure it.



(a)

Type	Status	Name
Shared	active	Optotun...
Shared	inactive	Optotune

Configuration name: Optotune
Status: not in use
Type: Shared configuration
Use as RTE startup configuration

(b)

Type	Status	Name
Shared	active	Optotune-B...
Shared	inactive	Optotune

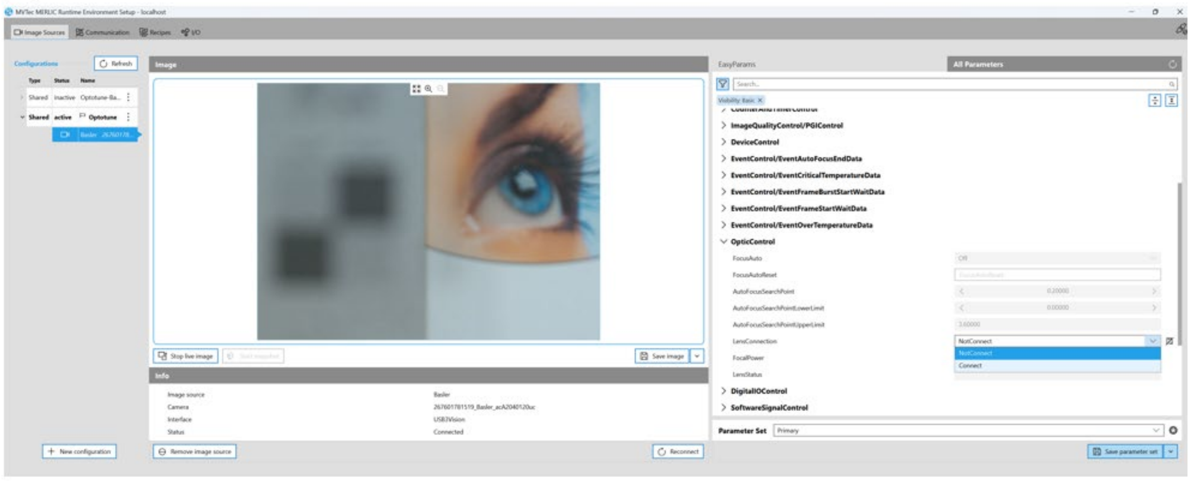
Name: Basler
Image Directory: Enter directory path
Device: Available devices
USB3Vision
267601781519_Basler_acA2040120uc

Figure 10. Image source selection in MERLIC

MERLIC provides a live view of the camera feed, allowing users to visualize the image in real time while adjusting camera parameters. All standard camera settings are accessible through the interface, including those for integrated liquid lenses. The specific control parameters for liquid lenses depend on the camera manufacturer and how they are organized within the GenICam feature tree. In the example shown using a Basler camera, these parameters are grouped under the **OpticControl** section. To enable control, the **LensConnection** parameter must first be set to **Connect**. Once the connection is established, the focal power of the lens can be adjusted interactively using the **FocalPower** slider, as illustrated in Figure 11(a). The AF can be executed using the FocusAuto command and its parameters are available in the AutoFunctionControl section, Figure 11(b).

Imagesource liveview can be added to the ToolFlow of MERLIC and the liquid lens features, such as any other parameters of the camera, can be adjusted in a sequence. The ability to control the liquid lens directly through camera parameters opens the door to developing a wide range of machine vision applications and tools within MERLIC.

(a)



Info	
Image source	Basler
Camera	267601781519_Basler_acA2040120uc
Interface	USB3Vision
Status	Connected

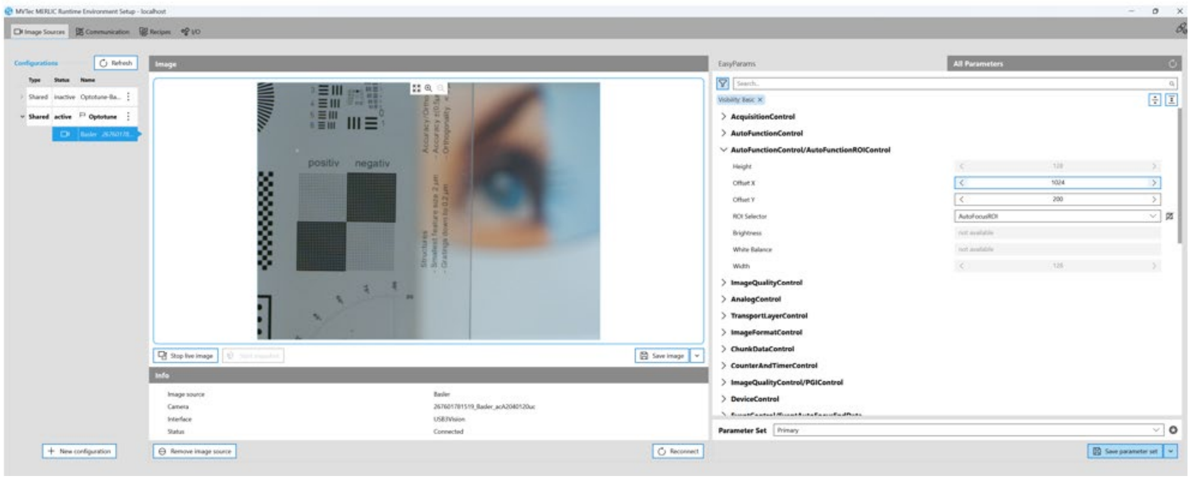
- ImageQualityControl/PGIControl
- DeviceControl
- EventControl/EventAutoFocusEndData
- EventControl/EventCriticalTemperatureData
- EventControl/EventFrameBurstStartWaitData
- EventControl/EventFrameStartWaitData
- EventControl/EventOverTemperatureData
- OpticControl

FocusAuto	Off
FocusAutoReset	
AutoFocusSearchPoint	0.20000
AutoFocusSearchPointLowerLimit	0.00000
AutoFocusSearchPointUpperLimit	3.60000
LensConnection	NotConnect (NotConnect / Connect)
FocalPower	
LensStatus	

- DigitalIOControl
- SoftwareSignalControl

Parameter Set: Primary

(b)



Info	
Image source	Basler
Camera	267601781519_Basler_acA2040120uc
Interface	USB3Vision
Status	Connected

- AcquisitionControl
- AutoFunctionControl
- AutoFunctionControl/AutoFunctionROIControl

Height	128
Offset X	1024
Offset Y	200
ROI Selector	AutoFocusROI
Brightness	not available
White Balance	not available
Width	128

- ImageQualityControl
- AnalogControl
- TransportLayerControl
- ImageFormatControl
- ChunkDataControl
- CounterAndTimerControl
- ImageQualityControl/PGIControl
- DeviceControl

Parameter Set: Primary

Figure 11. Camera parameters and liquid lens control in the image-source features

4 Conclusion

The integration of Optotune liquid lenses with modern machine vision systems offers a significant leap in flexibility, speed, and precision for industrial applications. By enabling electronic control of focus, these lenses eliminate the need for mechanical adjustments, reducing system complexity and increasing throughput. When combined with MVTec's HALCON and MERLIC software, users gain access to advanced image processing algorithms and easy-to-use tools that make implementing autofocus or depth adaptation straightforward.

The three interfacing approaches discussed—camera-integrated autofocus, HALCON-based control through the camera interface, and direct serial communication with the ECC-1C—cover a wide range of application requirements. Whether the goal is maximum simplicity with built-in camera firmware solutions or full control and customization using PC-based processing, Optotune lenses provide a scalable solution that fits various machine vision setups.

Ultimately, combining tunable optics, GenICam-compliant cameras, and powerful software platforms like HALCON and MERLIC unlocks highly adaptable, compact, and reliable systems. This synergy empowers diverse applications, from robotics and logistics to precision inspection and quality control, ensuring that machine vision solutions remain efficient, future-proof, and ready to meet the increasing demands of modern automation.