EN

Robot Vision

For Machine Vision Devices



Interface Protocol

Subject to change without notice
Available as PDF file only
Revision level: 13.04.2026
Version: 1.3.0
www.wenglor.com

Table of Contents

1. Use for Intended Purpose 3

2. Network Overview 3

3. Robot Vision Basics 4

4. Without wenglor Robot Server 6

4.1 Three-Point Calibration at Robot 8

4.2 Job in uniVision 9

4.3 Robot Program 12

5. With wenglor Robot Server 13

5.1 Basics with Robot Server 14

5.1.1 Camera on Robot 15

5.1.2 Camera not on Robot 20

5.2 Settings on Device Website 24

5.3 Calibration Job in uniVision 25

5.4 Detection Job in uniVision 26

5.5 Job in uniVision to get Target Pose or to calibrate Camera to Target 29

5.6 UR Robots with Polyscope 5 URCap 30

5.6.1 Installation of URCap 31

5.6.2 UR Installation 33

5.6.3 UR Program 40

5.7 UR Robots with Polyscope X Script 47

5.8 KUKA Robots 53

5.9 ABB Robots 57

5.10 Kassow Robots 62

5.10.1 Network Configuration of Robot Controller 62

5.10.2 Set Payload and TCP 63

5.10.3 Installation of CBUNs 63

5.10.4 Set up Workcell 67

5.10.5 Open Program and Setup Variables 68

5.10.6 Robot Program 70

5.11 Fanuc Robots 77

5.12 Generic Robot Vision Interfaces 88

5.13 Generic Robot Vision API 88

6. Change Index of Operating Instructions 95

1. Use for Intended Purpose

Machine Vision Devices can communicate directly to robots for robot vision applications (e.g. pick and place applications).
Supported Machine Vision Devices:

- Smart Camera B60
- Machine Vision Controller MVC with Machine Vision Cameras BBVK (recommended) and BBZK

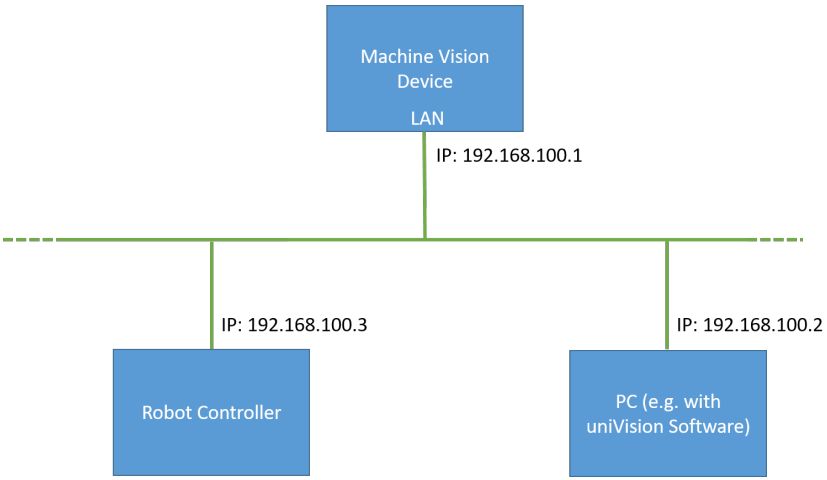
NOTE!
For details about the Machine Vision Devices check the operating instructions of the device. The interface protocol for Robot Vision addresses to users with knowledge in robotic programming and basic knowledge in Machine Vision.

2. Network Overview

Connect the network cable from the LAN port of the Machine Vision Device to the robot controller or to a switch in order to work with several devices.

NOTE!

- For details about the LAN port of the Machine Vision Device, check the operating instruction of the device.
- Cabling must be capable of 1 GBit/s throughout the entire network.
- Keep the networks for LAN, RTE and CAM separated in order to optimize the performance of the Machine Vision Device.
- Use a unique network configuration for the LAN port of the Machine Vision Device within the LAN network. Furthermore, the network configuration of the LAN port must be different to the network configuration of the RTE and CAM ports of the Machine Vision Device.



3. Robot Vision Basics

Machine Vision Devices can communicate with or without the wenglor robot server to the robot controller. The wenglor robot server runs on Machine Vision Devices and the settings are available on the device website of the Machine Vision Device. The robot server is available for each Processing Instance on the tab “Jobs” (for details, see operating instructions of Machine Vision Device).

- Without wenglor robot server (see section “4. Without wenglor Robot Server”):
  - » Communication to any robot manufacturer supporting TCP socket messaging.
  - » Camera on robot or not on robot.
  - » Mount the camera parallel to the measuring or picking plane.
  - » Three-point calibration on robot side to define the measuring or picking plane.
  - » Use Module Image Coordinate System within the uniVision job to align the coordinate systems of camera and robot.
  - » Use Module Image Calibration within the uniVision job to eliminate the lens distortion and to calculate in mm. Buy the calibration plate ZVZJ (recommended) or print corresponding PDF on flat and stiff material by yourself.
  - » Use Device TCP within the uniVision job for communication to robot controller via socket messaging

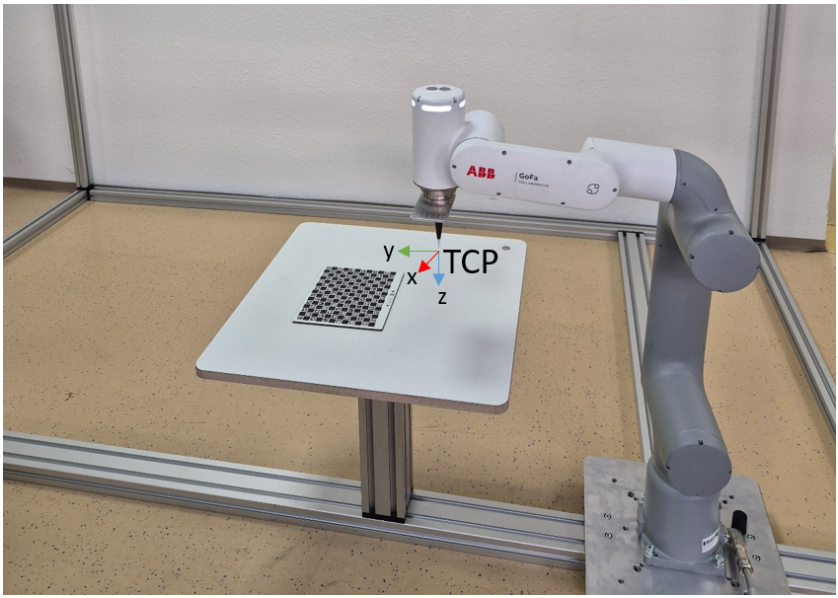


- With wenglor robot server (see section “5. With wenglor Robot Server”):
  - » Communication to supported robot manufacturers (listed in dropdown of Robot Manufacturer on the device website of the Machine Vision Device at the tab “Jobs”). Using "Generic" in the dropdown allows the communication to further robot manufacturers via the generic string based robot vision API (see section “5.12 Generic Robot Vision Interfaces”).
  - » Camera on robot or not on robot.
  - » If the camera is tilted towards the measuring or picking plane, the calibration compensates for this. Extreme angles should be avoided.
  - » Hand-eye calibration of robot and camera via several calibration poses where the camera looks from different positions and angles on the calibration plate. Buy the calibration plate ZVZJ (recommended) or print corresponding PDF on flat and stiff material by yourself.
  - » Calibration (including elimination of lens distortion and calculating in mm) is done within the robot server (no separate Module Image Calibration within the uniVision job is required).
  - » Use Device Robot Vision within the uniVision job for the communication to the robot server



The following basics are relevant for both cases:

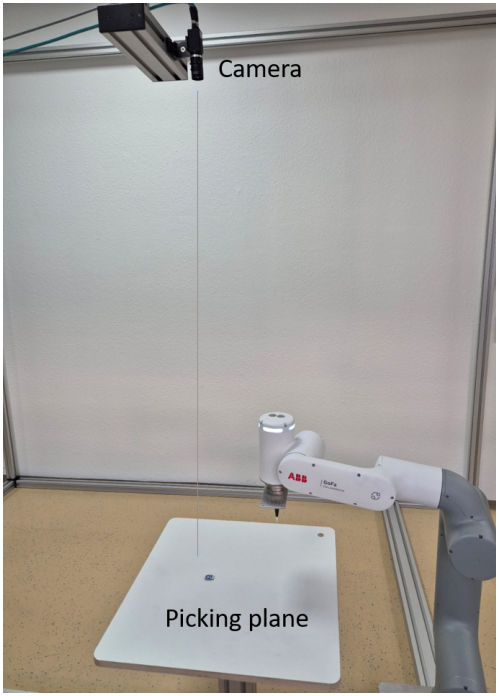
- Mount the camera on the robot or not on the robot. For details about mounting options, check the product detail page of the Machine Vision Device on www.wenglor.com.
- Make sure to use robot-compatible cables. For details, check the product detail page of the Machine Vision Device.
- For safe cable management, use the joint limits in the safety settings of the robot.
- Set the payload on the robot (if necessary).
- Teach the robot TCP (tool center point) in relation to the robot world coordinate system.



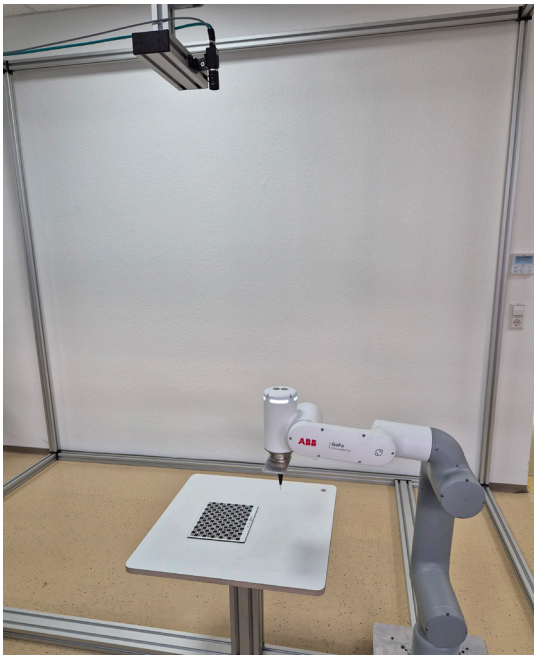
NOTE!
Make sure that the mechanical setup of camera, robot and picking plane is unchanged after the calibration. Otherwise, re-calibrating is necessary.

4. Without wenglor Robot Server

Without the wenglor robot server, the communication between robot and camera works via socket messaging for any robot manufacturer. Use one port for sending LIMA commands (e.g. to send trigger commands from the robot to the camera) and another port to receive process data from the camera (e.g. picking coordinates) via Device TCP.
It is possible to mount the camera on the robot or not on robot. It is important that the camera is parallel to the measuring or picking plane (tilted mounting is not supported). Three points define the picking plane for the robot.



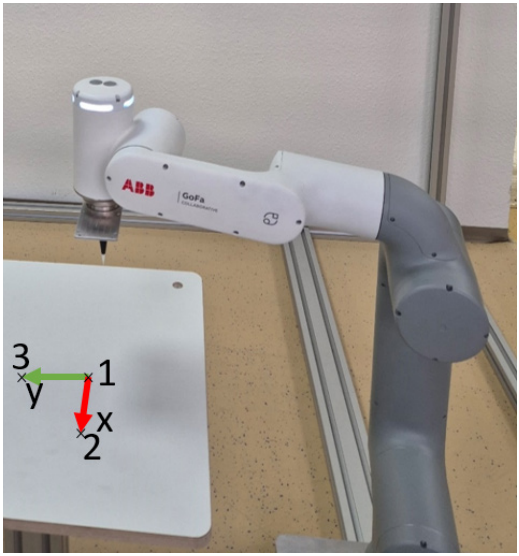
Make sure to align the coordinate systems of the robot and the camera via Module Image Coordinate System within the uniVision job. Use Module Image Calibration within the uniVision job to eliminate the lens distortion and to calculate the coordinates in mm. It requires one or several images of the calibration plate ZVZJ. Buy the product ZVZJ (recommended) or print the corresponding PDF on a flat and stiff material yourself. Use Device TCP to send the process data (e.g. coordinates of found object) to the robot.



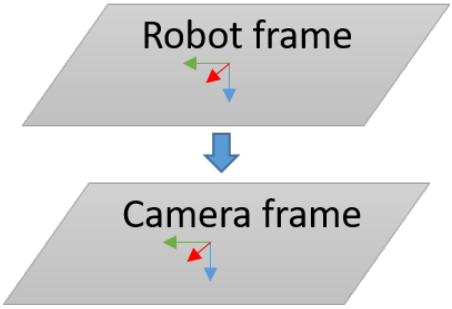
4.1 Three-Point Calibration at Robot

Calibrate the camera to the robot by creating a base frame on robot side that overlaps with the camera frame. Create the base frame at the robot via three points directly on the measuring or picking plane and mark it if needed to see it with the camera (make sure that the points are visible in the camera image):

1. Reference point
2. Point in x direction based on the reference point
3. Point in y direction based on the reference point

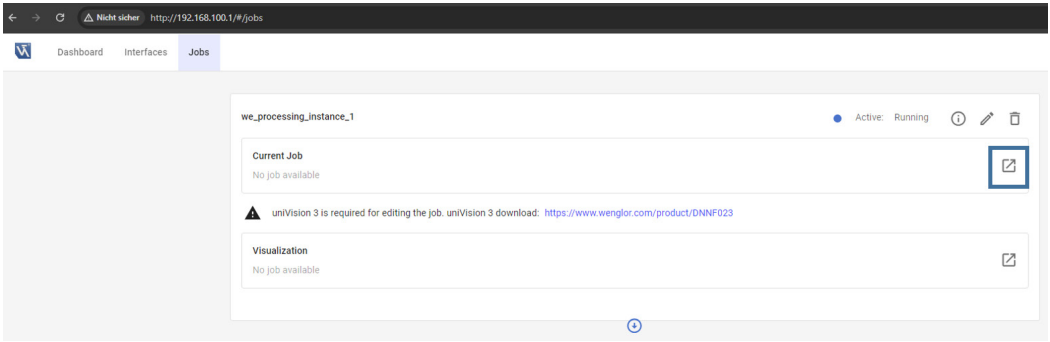


NOTE!
Make sure to use the same origin and the same orientation for robot and camera frame. Module Image Coordinate System defines the coordinate system of the camera (see section “4.2 Job in uniVision”).



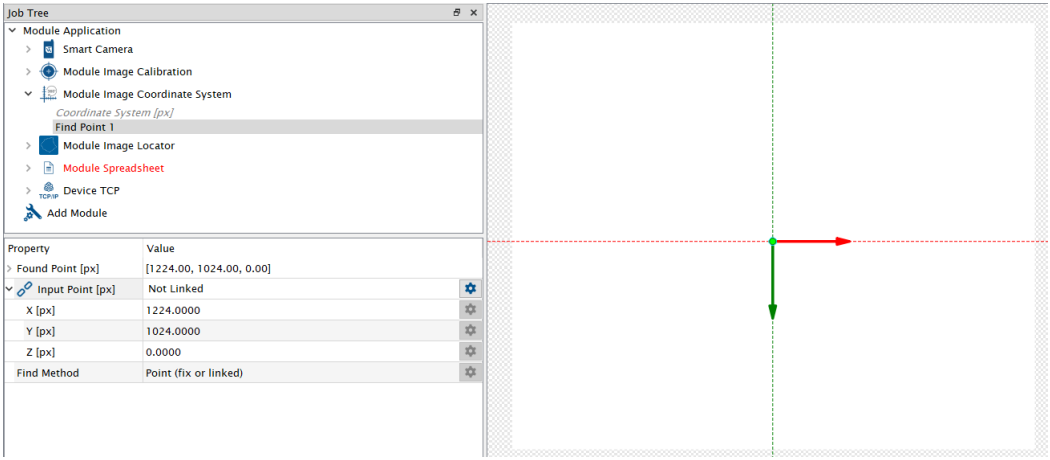
4.2 Job in uniVision

Open the device website of the Machine Vision Device (via entering the IP address in the browser, by default 192.168.100.1), access the tab Jobs and open the current job in the uniVision 3 software.

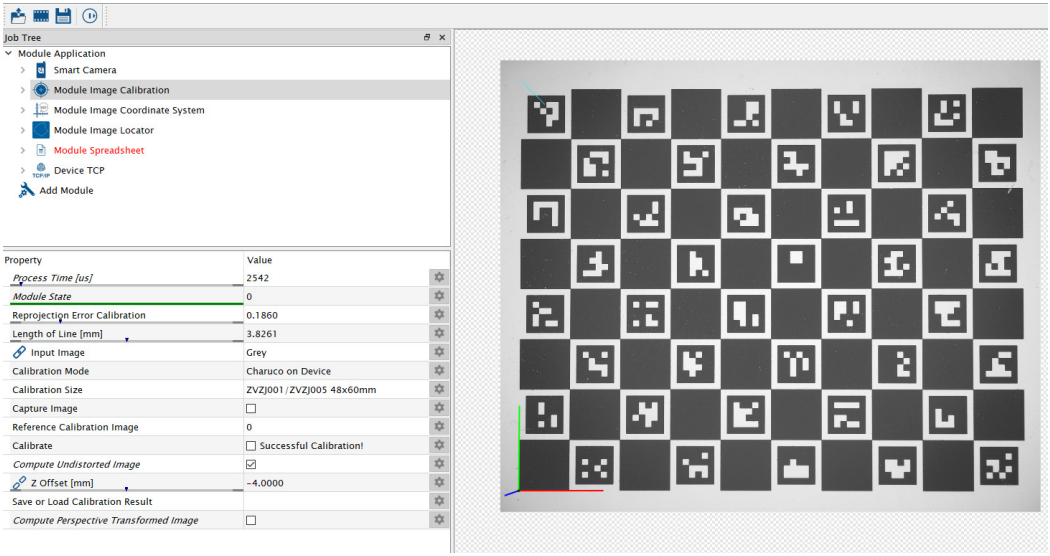


Load the template “Pick objects with generic robot” for easy setup.
Adjust focus and brightness of the camera to get a sharp and well illuminated image. By default, Trigger Mode is set to On and Trigger Source to Software (mandatory for picking). Set Trigger Mode to Off in order to adjust the camera image. Afterwards change the setting back to Software trigger.

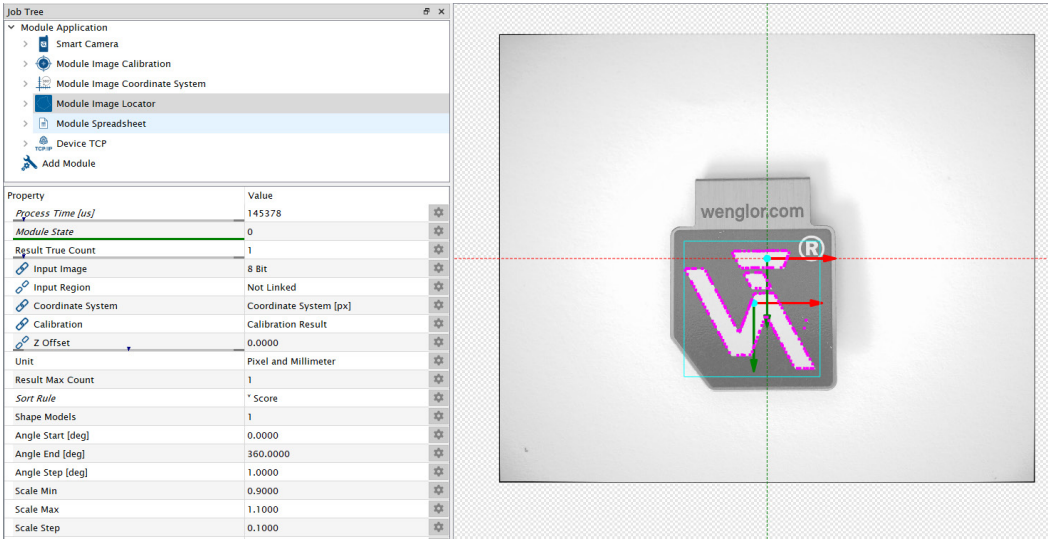
Use Module Image Coordinate System in order to create a common frame between camera and robot. By default, the origin of the coordinate system is in the center of the camera image (if using a 5 MP camera). Use the same coordinate system at the robot (see section “4.1 Three-Point Calibration at Robot”). Use your markings on the picking plane to adjust the coordinate system.



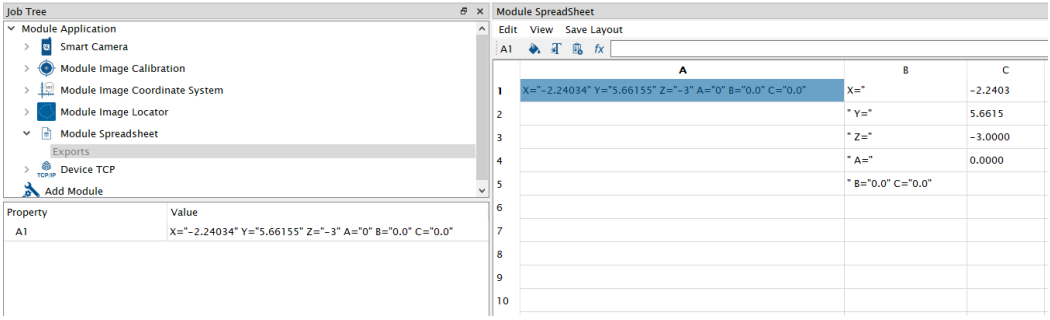
Use Module Image Calibration to eliminate the lens distortion and to calculate the coordinates in mm. Buy the calibration plate ZVZJ (recommended) or print the corresponding PDF yourself on a flat and stiff material. Put the calibration plate in the field of view of the camera. Select the corresponding size of the calibration plate ZVZJ at Module Image Calibration, click on “Capture Image” and afterwards on “Calibrate”. The Z Offset of -4 mm compensates the height of the calibration plate ZVZJ. Adjust it if the height of your calibration plate is different.



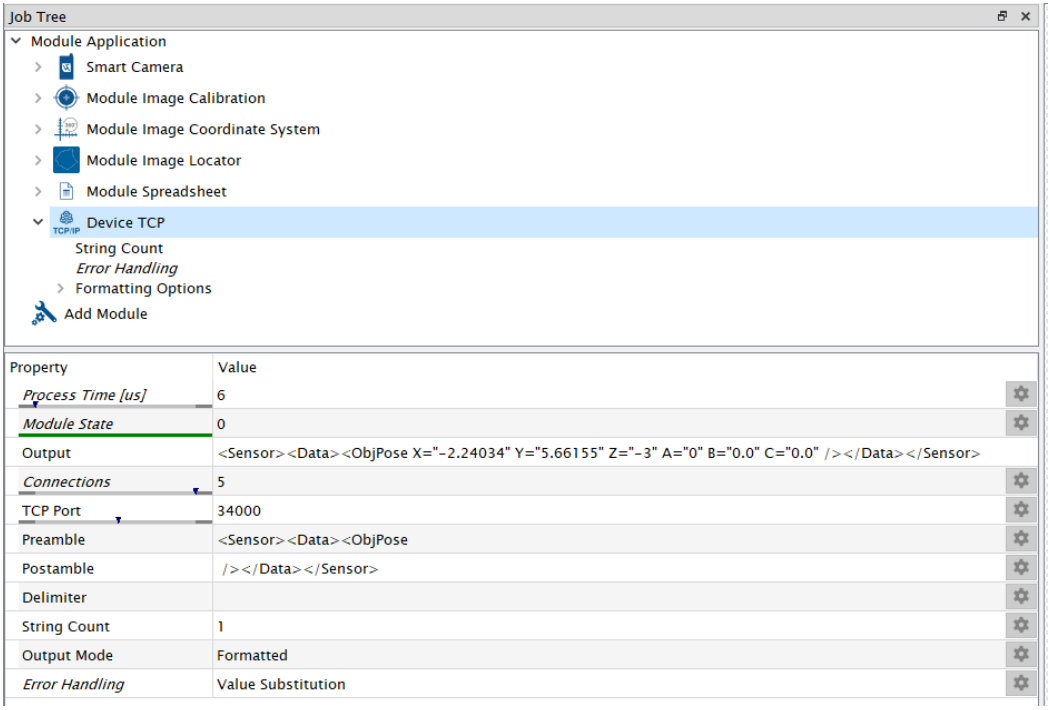
Put your picking object below the camera and teach it in Module Image Locator (at sub-module “Shape Models”). Then, Module Image Locator finds the object according to the parameters.



Use Module Spreadsheet to create one string with the coordinates x, y and z and the rotations A, B and C. Enter the height of the picking object at Z. The following example works with robots based on XML communication (tested with KUKA robots). In general, the robot programmer and the user configuring the Machine Vision Device must agree on a common protocol.



Use Device TCP to send the result via socket messaging to the robot.



NOTE!
For details about uniVision 3, check the separate operating instructions of the software wenglor uniVision 3 (DNNF023).

Save the job in the projects folder of the Machine Vision Device.

4.3 Robot Program

The communication from the robot to the camera via socket messaging is based on the TCP/IP protocol. Create two socket connections from the robot:

- One socket connection to the LIMA Read Write Limited port for trigger commands (based on request/answer).
- One socket connection to Device TCP in order to get the process data (the camera sends results after each trigger without requests).

NOTE!
For details, check the network interfaces chapter at the operating instructions of the software wenglor uniVision 3 (DNNF023).

The pseudo code on robot side looks as follow:

- Create the LIMA Read Write Limited socket.
- Connect the LIMA Read Write Limited socket to the IP address (by default 192.168.100.1) and the port (e.g. 33060 for the first processing instance) of the Machine Vision Device.
- Create the Device TCP socket.
- Connect the Device TCP socket to the IP address (by default 192.168.100.1) and the Device TCP port (e.g. 34000 for the first processing instance) of the Machine Vision Device.
- Send LIMA trigger command <T/> through LIMA Read Write Limited socket.
- Wait for the response from LIMA through LIMA Read Write Limited socket.
- Read the result from Device TCP socket.
- Move the robot to the coordinates provided by the Device TCP socket.

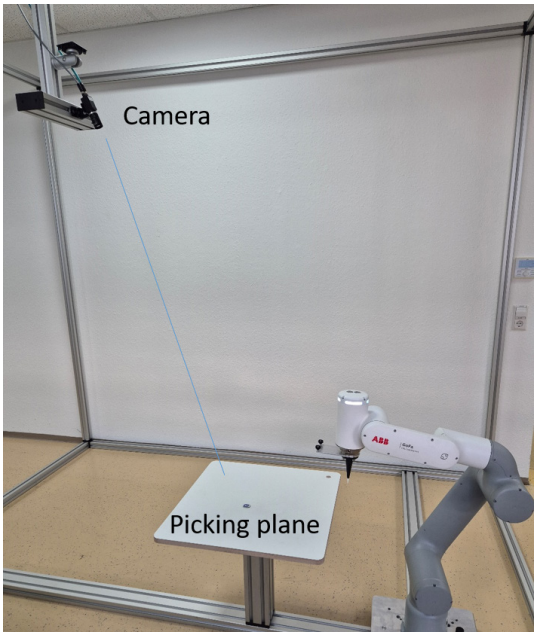
NOTE!

- In case of camera on robot, make sure that the robot position when triggering the camera is the same position that was used for calibrating the camera via Module Image Calibration.
- For details about Device TCP and LIMA commands, check the operating instructions of the software wenglor uniVision 3 (DNNF023).

5. With wenglor Robot Server

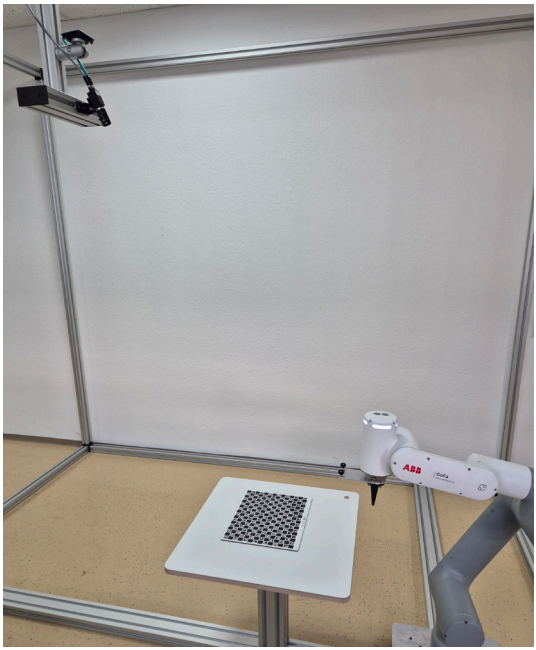
The wenglor robot server supports the communication to specific robot manufacturers. For details, check the dropdown list of robot manufacturers on the device website of the Machine Vision Device and the operating instructions of the device. Using "Generic" in the dropdown allows the communication to further robot manufacturers via the generic string based robot vision API (see section “5.12 Generic Robot Vision Interfaces”).

It is possible to mount the camera on the robot or not on the robot. Even slightly tilting of the camera towards the measuring or picking plane is supported.



It is also possible to mount camera and robot on a mobile platform (e.g. on an Automated Guided Vehicle - AGV) in order to correct the positional deviation of the mobile platform in front of the machine or the shelf.

Calibrate robot and camera via several calibration poses where the camera looks from different positions and angles on the calibration plate (hand-eye calibration). Buy the calibration plate ZVZJ (recommended) or print the corresponding PDF on a flat and stiff material. The calibration is done inside of the wenglor robot server including the compensation of the lens distortion and the calculations in mm (Module Image Calibration is not needed within the uniVision job). Use Device Robot Vision within the uniVision job to send results (coordinates of found objects) to the robot server.



5.1 Basics with Robot Server

The calibration process is different, if the camera is mounted on the robot or not on the robot.
Buy one of the calibration plates ZVZJ (see wenglor website https://www.wenglor.com → ZVZJ) or print the PDF on a stiff and flat material (see https://www.wenglor.com → ZVZJ). For best accuracy, use the wenglor calibration plates ZVZJ.

NOTE!

- In case of printing, make sure to print the PDFs at actual size.
- Typically, the reprojection error for the ZVZJ calibration plate is five times smaller compared to the printed version.
- For direct light applications, non-transparent calibration plates made of carbon fiber are available. For backlight applications, transparent calibration plates with the material glass are available.
- The calibration plate should cover at least half of the image and should be visible completely by the camera if possible for most accurate results.

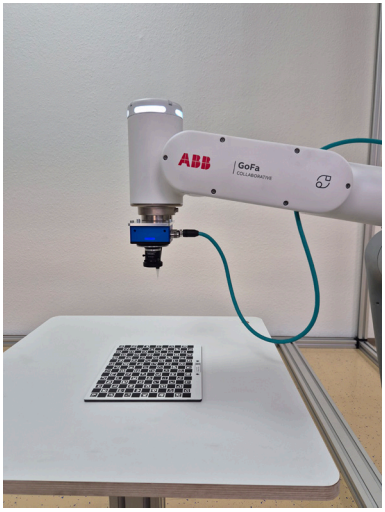
Consider the following points when setting the calibration poses:

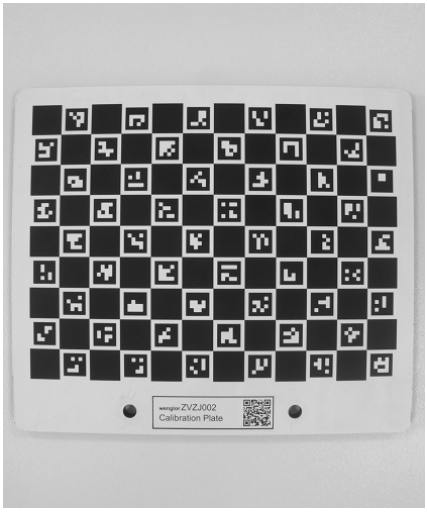
- Check that the camera sees the calibration plate when setting the poses.
- Make sure that the difference between one pose and the next one is as big as possible for best accurate results (motions with non parallel rotation axes are required so make sure to variate with the pose angles as much as possible). The same pose could be used several times if not consecutive (e.g. pose one and three can be similar). If space is limited in the application, small differences between one pose and the next one are also possible, but result in less accurate results.
- The calibration plate should cover as much as possible of the camera image and should be visible completely if possible for most accurate results.

5.1.1 Camera on Robot

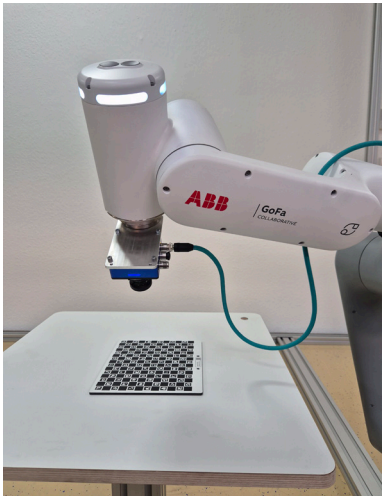
Calibration use case

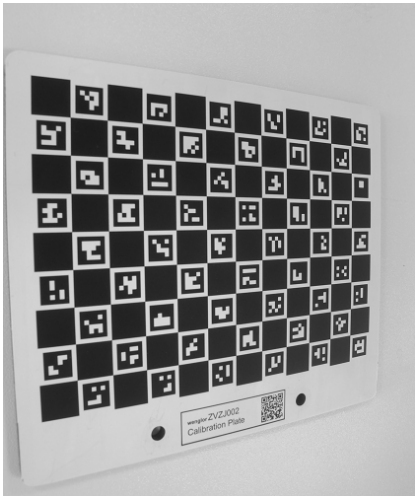
If the camera is on the robot, teach minimum five calibration poses. For more accurate results, teach further calibration poses (e.g. seven to eleven poses).
The first calibration pose is also the detection pose. Make sure to use a suitable pose where the objects are located in a safe way.



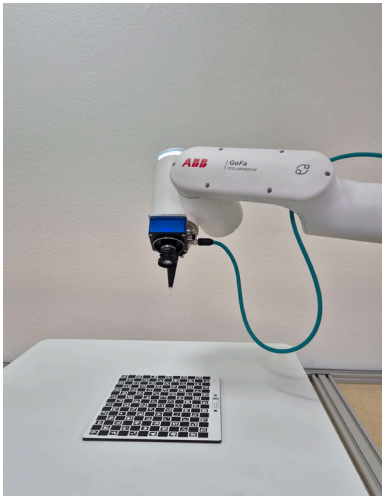


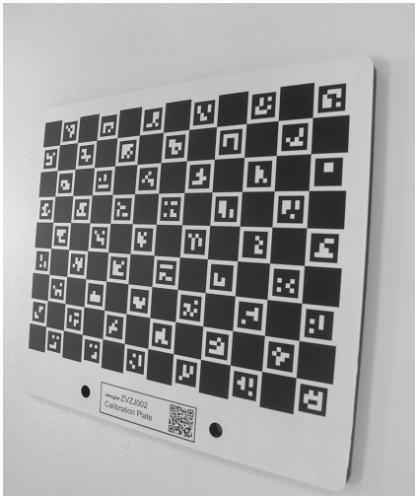
Make sure to use big variations between the different poses (especially variations in the pose angles). See examples for four further calibration poses.



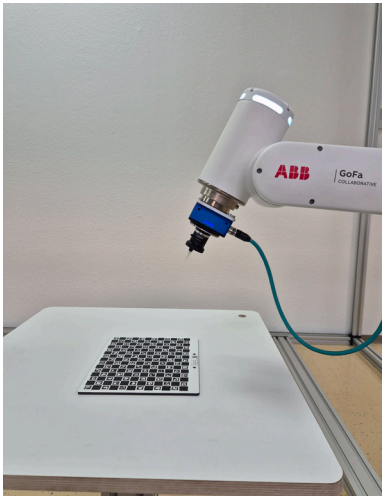


Calibration pose 2

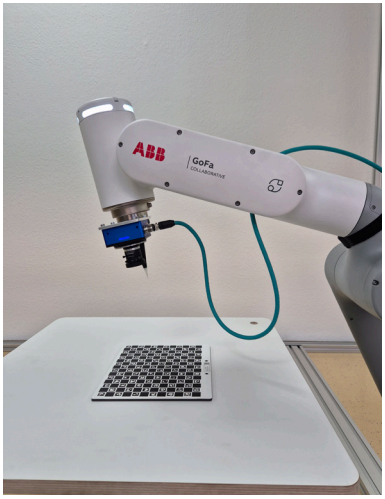




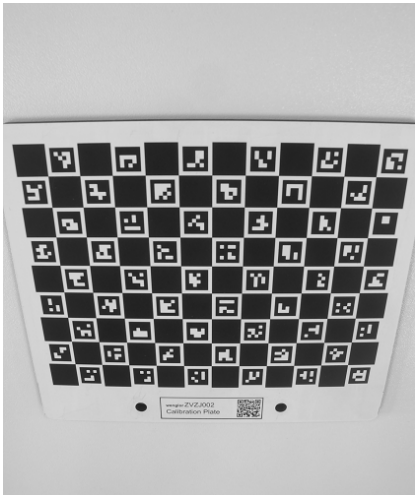
Calibration pose 3



Calibration pose 4



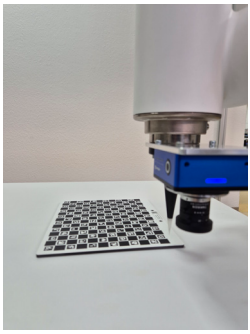
Calibration pose 5



After calibration, an optional verification step can be performed to check its accuracy. Applying it, moves the robot TCP to the bottom left corner of the calibration plate (with an adjustable safety height offset). It is necessary that the calibration plate was not moved between the calibration and the verification step. In case of bad results, check the setup and re-run the calibration.

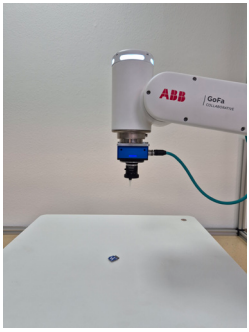
NOTE!

- For the verification step, the Z-axis must point to the object plane.
- The reprojection error shows how good the calibration was. Typical values are 0.1 for ZVZJ calibration plates and 0.5 for printed calibration plates.



Detect use case

After successful calibration, pick your objects. For the detection pose, it is mandatory to use the position of the first calibration pose.



With the object position sent by the camera, the robot moves to the object pose.



Mobile platforms and further use cases
(e.g. to correct positional deviations of mobile platforms in front of the machine or shelf)

If using camera and robot on a mobile platform, make sure that the calibration plate is mounted fix at each machine or shelf as reference position. Run once in the beginning the normal calibration process with several different poses with big variations (especially differences in the pose angles). Afterwards in the run use case, when the mobile platform is in front of the machine or shelf, the camera captures only one image of the calibration plate and calculates the positional deviation of the mobile platform towards the target position. Use this info to update the reference frame of the machine or shelf if the handling pose is fix.



In case of variating handling poses at mobile platforms, it is possible to calibrate the camera to target relation and afterwards to use the detect command. Run once in the beginning the normal calibration process with several different poses with big variations (especially differences in the pose angles). Afterwards in the run use case, the camera captures only one image of the calibration plate and then it captures another image via the detect command for flexible object picking. Make sure that the position of the mobile platform is unchanged between calibrating the camera to the calibration target and detecting the object and that the same robot pose is used for calibrating the camera to the target and for detecting the objects.

NOTE!

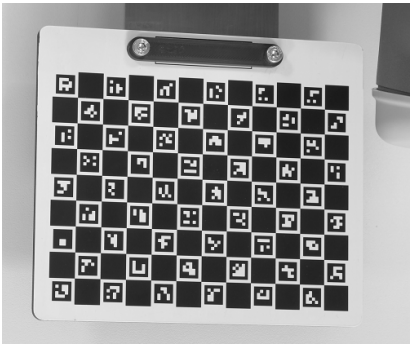
- It is also possible to mount camera and robot in the machine. Then the calibration plate must be at a fix reference position on the mobile platforms or on the boxes.
- For unique identification of the machine, the shelf or the mobile platform, it is possible to label them e.g. with codes. The camera can read the code and send it to the robot as additional info. The code result must be linked to the additional value in Device Robot Vision in the uniVision job.

5.1.2 Camera not on Robot

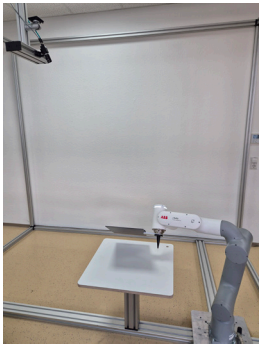
Calibration use case

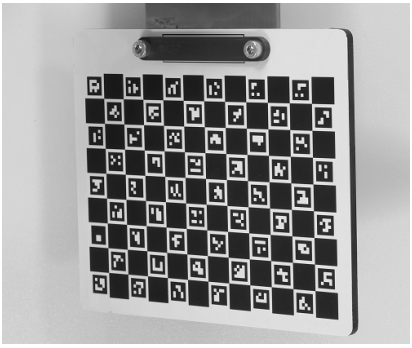
If the camera is not on the robot, the calibration process consists of two steps.
For the first step of the calibration process, mount the calibration plate on the robot. Teach minimum five calibration poses (e.g. seven to eleven poses). For more accurate results, teach further calibration poses. Make sure to use big variations between the different poses (especially variations in the pose angles).



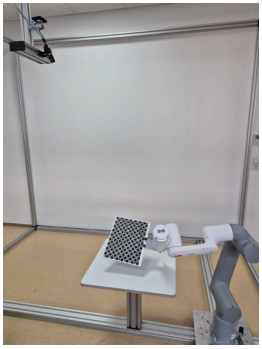


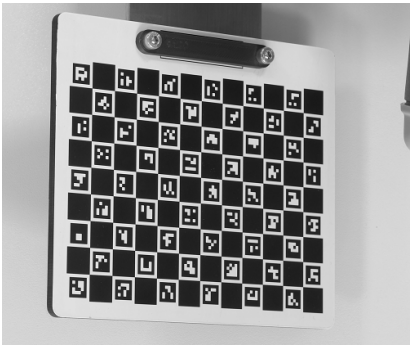
Calibration pose 1



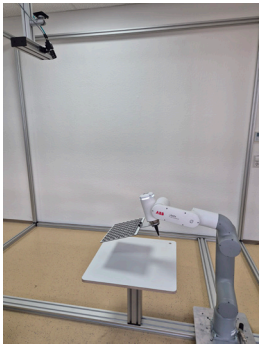


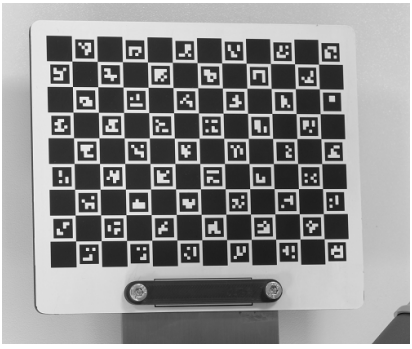
Calibration pose 2



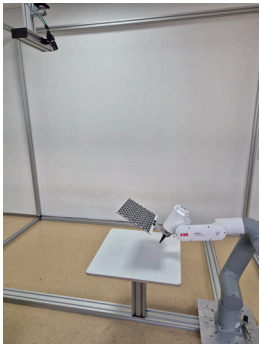


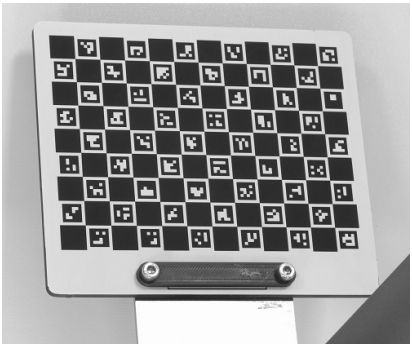
Calibration pose 3





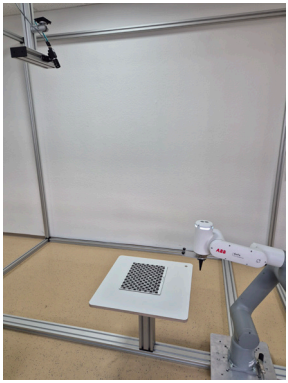
Calibration pose 4





Calibration pose 5

For the second step of the calibration process, put the calibration plate on the measuring or picking plane and capture one single image.



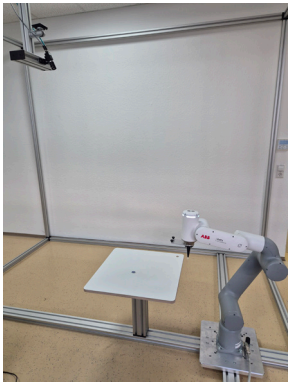
After running the calibration, an optional verification step is possible to check the accuracy of the calibration. Applying it, moves the robot TCP to the bottom left corner of the calibration plate (with an adjustable safety height offset). It is necessary that the calibration plate was not moved between the second calibration step and the verification step. In case of bad results, check the setup and re-run the calibration.

NOTE!

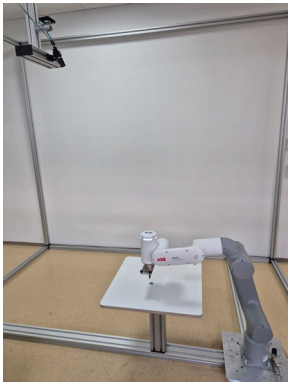
- For the verification step, the z axis must point to the object plane. In tilted settings, the Z-axis of the TCP will be perpendicular to the object plane.
- The reprojection error shows how good the calibration was. Typical values are 0.1 for ZVZJ calibration plates and 0.5 for printed calibration plates.

Detect use case

After successful calibration, pick your objects. Use any detection pose where the camera sees the objects.



With the object position sent by the camera, move the robot to the object.



Mobile platforms and further use cases
(e.g. to correct positional deviations of mobile platforms in front of the machine or shelf)

If using camera and robot on a mobile platform, make sure that the calibration plate is mounted fix at each machine or shelf as reference position. Run once in the beginning the normal first calibration step with several different poses with big variations (especially differences in the pose angles) when the calibration plate is mounted on the robot. Also run once the second calibration step, e.g. when the camera sees the calibration plate at the machine. Afterwards in the run use case, when the mobile platform is in front of the machine or shelf, the camera captures only one image of the calibration plate and calculates the positional deviation of the mobile platform towards the target position. Use this info to update the reference frame of the machine or shelf if the handling pose is fix.



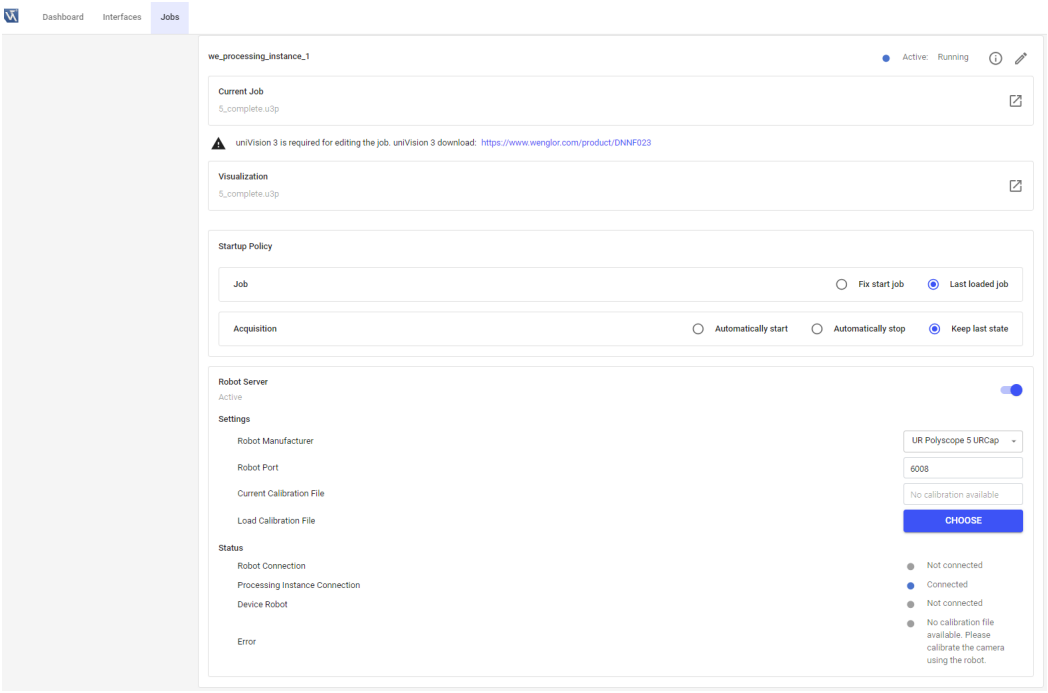
In case of variating handling poses at mobile platforms, it is possible to calibrate the camera to target relation and afterwards to use the detect command. Run once in the beginning the normal first calibration step with several different poses with big variations (especially differences in the pose angles). Also run once the second calibration step, e.g. when the camera sees the calibration plate at the machine. Afterwards in the run use case, the camera captures only one image of the calibration plate and then it captures another image via the detect command for flexible object picking. Make sure that the position of the mobile platform is unchanged between calibrating the camera to the calibration target and detecting the object.

NOTE!

- It is also possible to mount camera and robot in the machine. Then the calibration plate must be at a fix reference position on the mobile platforms or on the boxes.
- For unique identification of the machine, the shelf or the mobile platform it is possible to label them e.g. with codes. The camera can read the code and send it to the robot as additional info. The code result must be linked to the additional value in Device Robot Vision in the uniVision job

5.2 Settings on Device Website

Open the device website of the Machine Vision Device and select the tab “Jobs” to see the robot server settings for each processing instance.

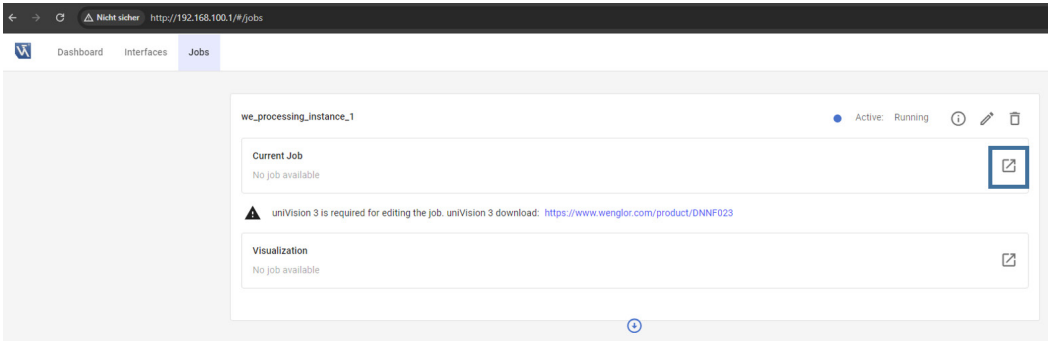


By default, the robot server is active on the Smart Camera B60 and inactive on the Machine Vision Controller MVC. Activate the robot server, if needed. Additional parameters appear if activated:

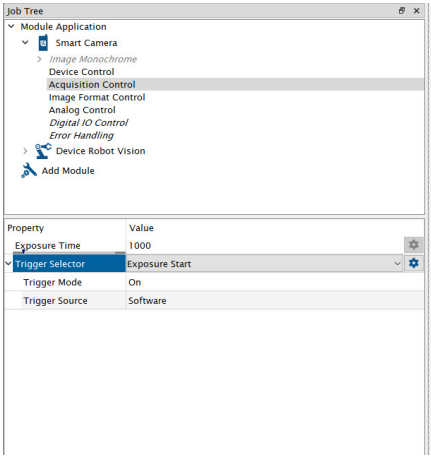
- Settings
  - » Robot Manufacturer: Select the relevant robot manufacturer (UR Polyscope 5 URCap, UR Polyscope X Script, Kuka, ABB, Kassow Robots, Fanuc or Generic).
  - » Robot Port: Depending on the robot manufacturer, the default port is shown. Edit the port, if needed (e.g. in case of port conflicts at several processing instances on the Machine Vision Controller MVC). Make sure to use a unique port. In case the port is already in use, change the port and wait approximately one minute for the error to disappear.
  - » Current Calibration File: Shows the name of the currently loaded calibration file.
  - » Load Calibration File: Load another calibration file. The calibration file is only valid for one specific device and valid only if the relation between camera, lens and robot is unchanged (e.g. it gets invalid if the camera position is changed or if the lens is changed, e.g. via another focus position at B60 auto-focus devices).
- Status:
  - » Robot Connection: Shows if the robot controller is connected to the robot server or not.
  - » Processing Instance Connection: Shows if the robot server is connected to the processing instance via the LIMA Read Write Limited port or not.
  - » Device Robot: Shows if the robot server is connected to Device Robot Vision or not.
  - » Error: Shows if there is any error (e.g. if calibration file does not fit to the device).

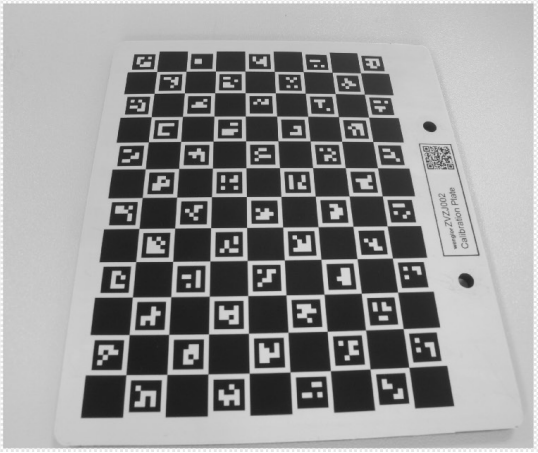
5.3 Calibration Job in uniVision

Open the device website of the Machine Vision Device, access the tab Jobs and open the current job in the uniVision 3 software.



Load the template "Calibrate camera with robot" for easy setup.





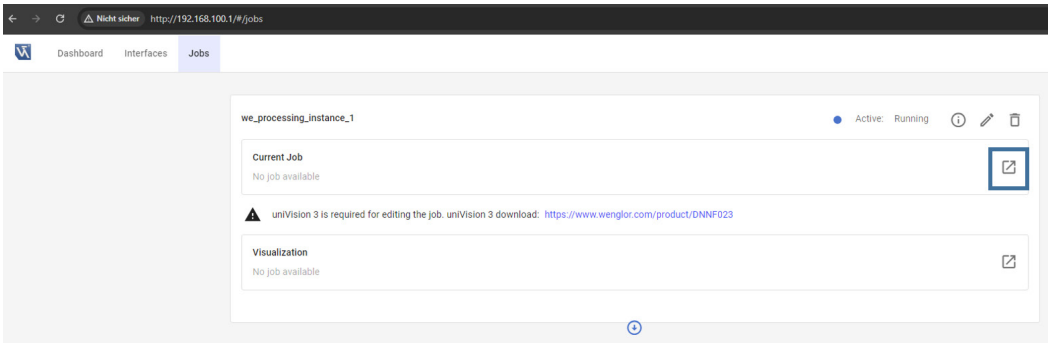
Adjust focus and brightness of the camera to get a sharp and well illuminated image. By default, Trigger Mode is set to On and Trigger Source to Software (mandatory for the calibration procedure). Adjust Trigger Mode to Off and switch to Run Mode in order to adjust the camera image. Afterwards change the setting back to Software trigger. If working with color images, make sure that “Create BGRA Image” is active at the input camera (only relevant if working with URCap).

Make sure that Device Robot Vision with its default settings is part of the uniVision job.

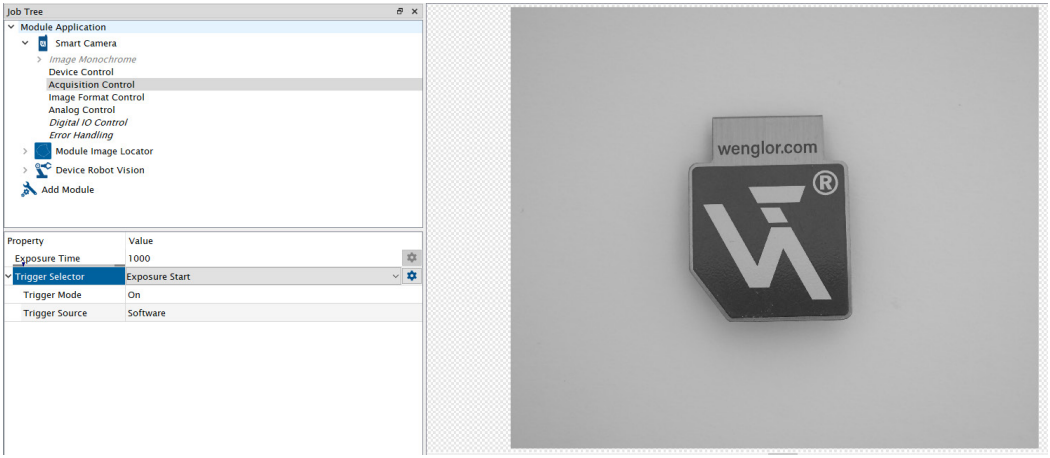
Save the calibration job in the device projects folder on the Machine Vision Device so that the robot can load it later.

5.4 Detection Job in uniVision

Open the device website of the Machine Vision Device, access the tab Jobs and open the current job in the uniVision 3 software.

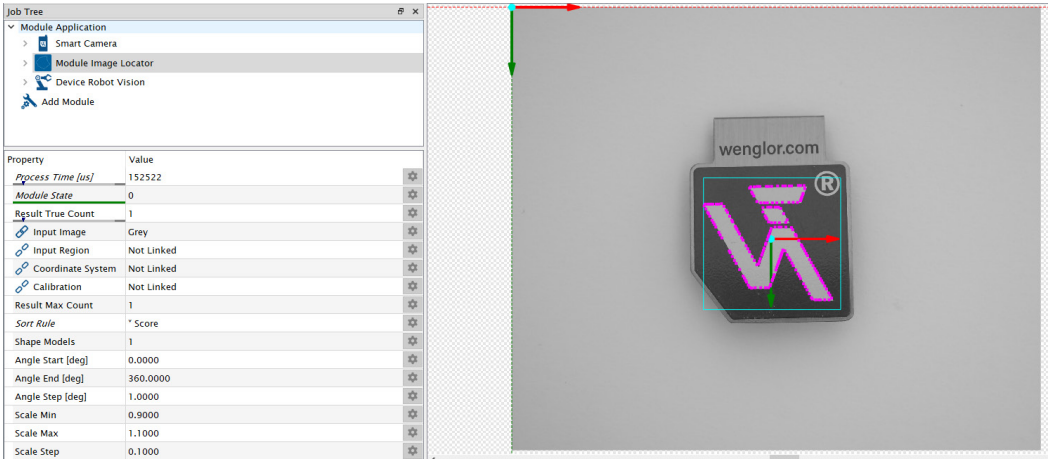


Load the template "Pick objects with robot" for easy setup.



Adjust focus and brightness of the camera to get a sharp and well illuminated image. By default, Trigger Mode is set to On and Trigger Source to Software. Adjust Trigger Mode to Off and switch to Run Mode in order to adjust the camera image. Afterwards change the setting back to Software trigger. If working with color images, make sure that “Create BGRA Image” is active at the input camera (only relevant if working with URCap).

Teach your object at Module Image Locator (or Module Image Pattern Match). For details, check the operating instructions of the software wenglor uniVision 3 (DNNF023).



In the template, Device Robot Vision is already pre-configured to send the results to the robot server.

Process Time [µs]	Process time to run the module in µs
Module State	Shows state of module: • 0: No error • Different to 0: Error
Output	Shows preview of the output sent to the wenglor robot server.
Shape Model Count	Defines the number of shape models (object types) used in the sub-module "Shape Model Height Difference"
Result Max Count	Defines the number of results used in the sub-module "Result List".
Result True Count	Link the value with Result True Count of Module Image Locator or Module Image Pattern Match so that the robot knows how many objects are within the current image. By default, the value is 0. Linking Result True Count is mandatory so that the robot server knows how many objects are found in the current image.

Sub-Module “Shape Model Height Difference” (number of entries depends on parameter Shape Model Count):

- Height Difference to Calibration [mm] #...x: Defines the height difference of each shape model to the calibration plate. By default, the value is 0.

NOTE!

- It is possible to pick different object types with different heights.
- Enter the object height relative to the surface of the calibration plate in mm. If the object is lower than the calibration plate plane, enter a negative number. If the object is higher, enter a positive number. Use the height difference also in case of height variations after the calibration process.

Sub-Module “Result List” (number of entries depend on parameter Result Max Count)

Link the result list of Module Image Locator or Module Image Pattern Match to the Result List of Device Robot Vision.

Shape Model	Link the shape model (object type) for each result. By default, the value is 0.
X [px]	Link the x position for each result. Linking the x coordinate is mandatory.
Y [px]	Link the y position for each result. Linking the y coordinate is mandatory.
Phi (Z-Rotation) [deg]	Link the Phi value (z rotation) for each result. Linking Phi is mandatory.
Additional Value	Optionally link an additional string result (e.g. score value) for each result.

NOTE!

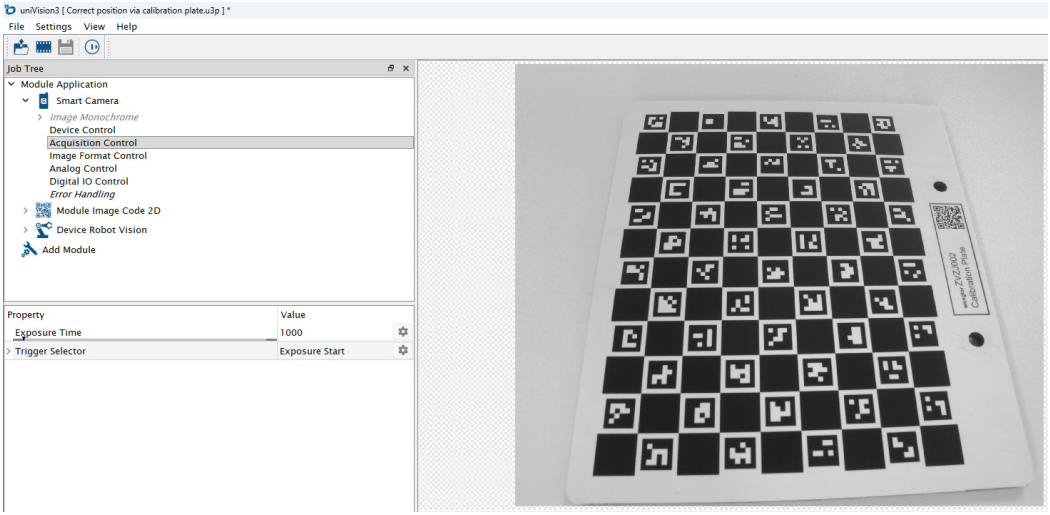
- It is possible to detect and pick multiple objects in a single image capture (if setup accordingly in the uniVision job).
- Offsets in x and y can be set directly in Module Image Locator or Module Image Pattern Match.
- Pixel results for x and y can only be used by the robot if created by modules that use the original camera image or any image based on it. No support of x and y coordinates created by modules that use the undistorted or perspective transformed image of Module Image Calibration or any image based on it. Exceptionally, it is supported to use the undistorted image in Module Image Locator or in Module Image Pattern Match with linked input calibration if linking the distorted pixel coordinates at Device Robot Vision.

Save the detect objects job in the device projects folder of the Machine Vision Device so that the robot can load it later.

5.5 Job in uniVision to get Target Pose or to calibrate Camera to Target

In case of camera and robot mounted on a mobile platform, load the template "Correct position via calibration plate".

Adjust focus and brightness of the camera to get a sharp and well illuminated image. By default, Trigger Mode is set to On and Trigger Source to Software. Adjust Trigger Mode to Off and switch to Run Mode in order to adjust the camera image. Afterwards change the setting back to Software trigger. If working with color images, make sure that “Create BGRA Image” is active at the input camera (only relevant if working with URCap).

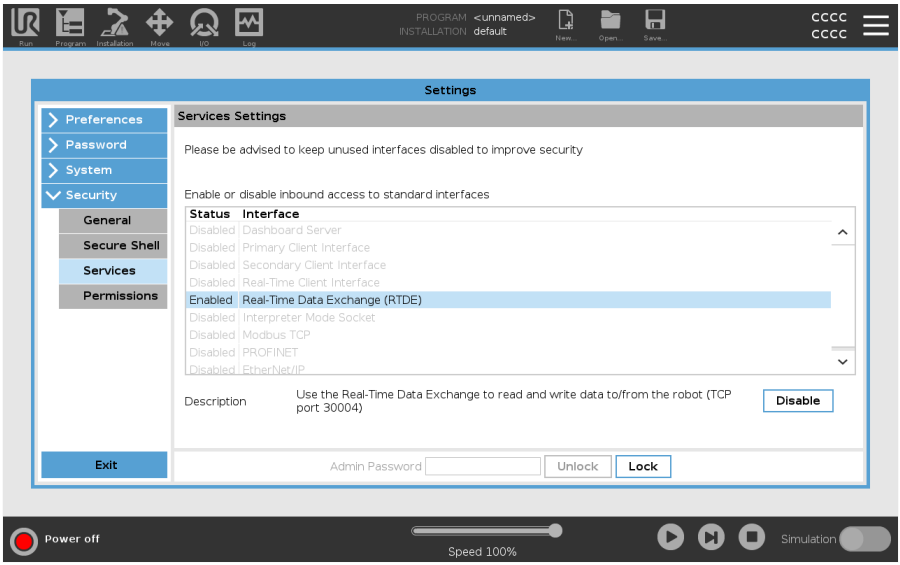


Make sure that Device Robot Vision is part of the uniVision job. Optionally, link any job result as "Additional Value" in Device Robot Vision (Result List → 0) to identify the current position (e.g. via linking a code result). Make sure that "Shape Model" of result "0" in the Result List is set to a valid value (e.g. "0") and that "Result True Count" is linked with the corresponding value (e.g. "Result True Count" of the code module).

Save the job in the device projects folder on the Machine Vision Device so that the robot can load it later.

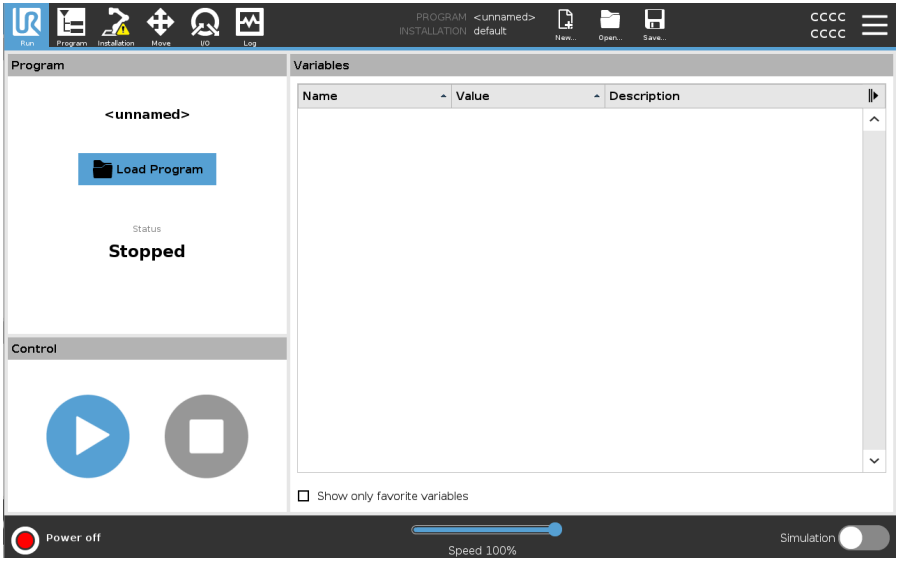
5.6 UR Robots with Polyscope 5 URCap

Supported UR robots: E-series with Polyscope version 5 (CB-series of UR is not supported).
Make sure that Real Time Data Exchange (RTDE) is enabled in the security settings of UR. The minimum supported Polyscope version is 5.22.

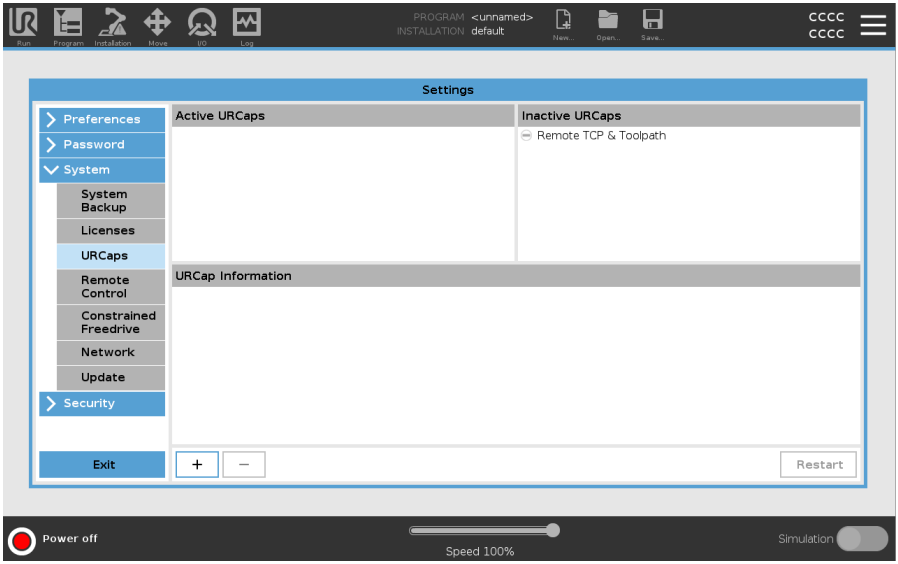


5.6.1 Installation of URCap

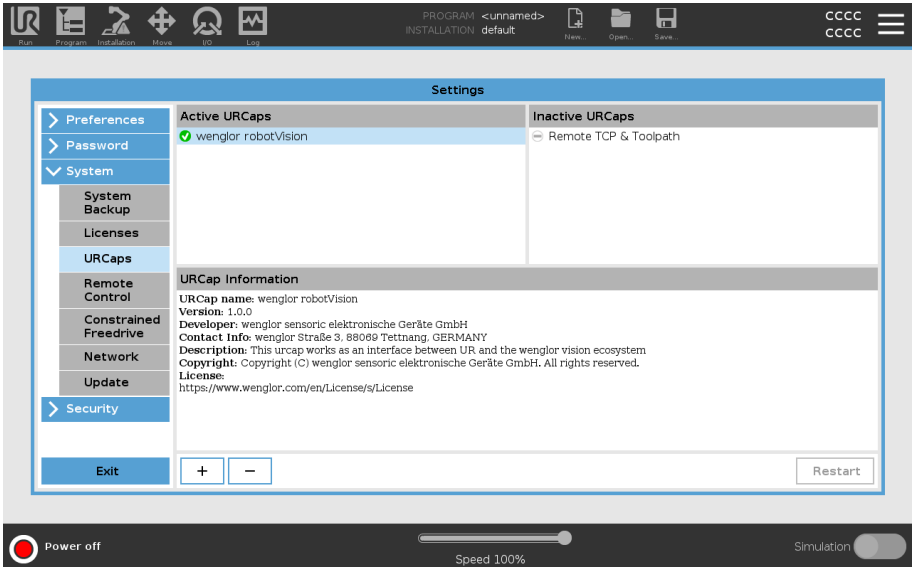
Download the latest URCap version from https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision and put the file on a freshly formatted stick.
Plug in the USB stick into the UR Teach Panel in order to install the URCap. Start the robot and go to the menu in the top right corner of the UR Teach Panel.



Navigate then to the System menu and select URCaps. Click on the + Symbol



Navigate through the system until you find the URCap file “wenglor robot vision”. Select it and proceed by re-starting the robot. After the reboot, the URCap is displayed with a green hook after the successful installation.

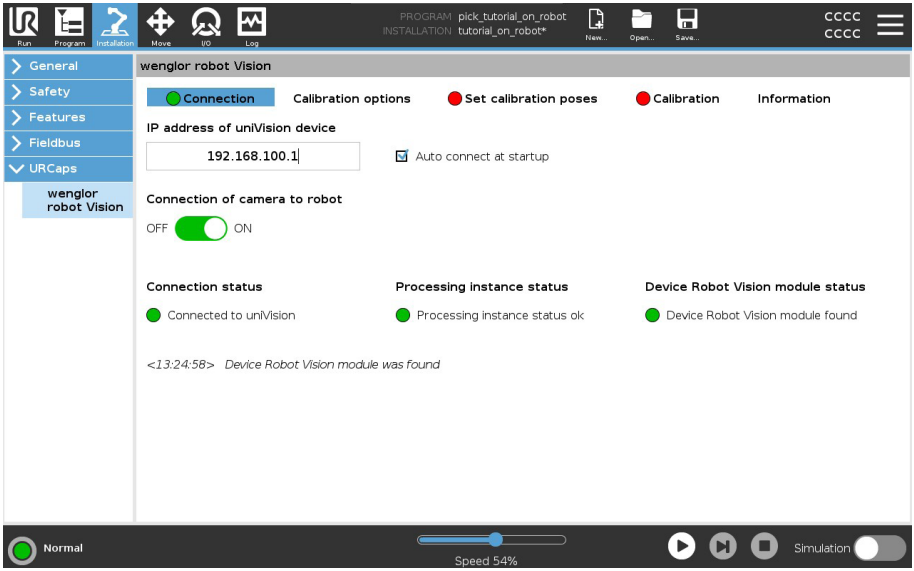


5.6.2 UR Installation

The installation enables to connect to the Machine Vision Device and to calibrate camera and robot. Select Installation on the UR Teach Panel and click on “URCaps” → “wenglor robot vision”.

LED status in tabs

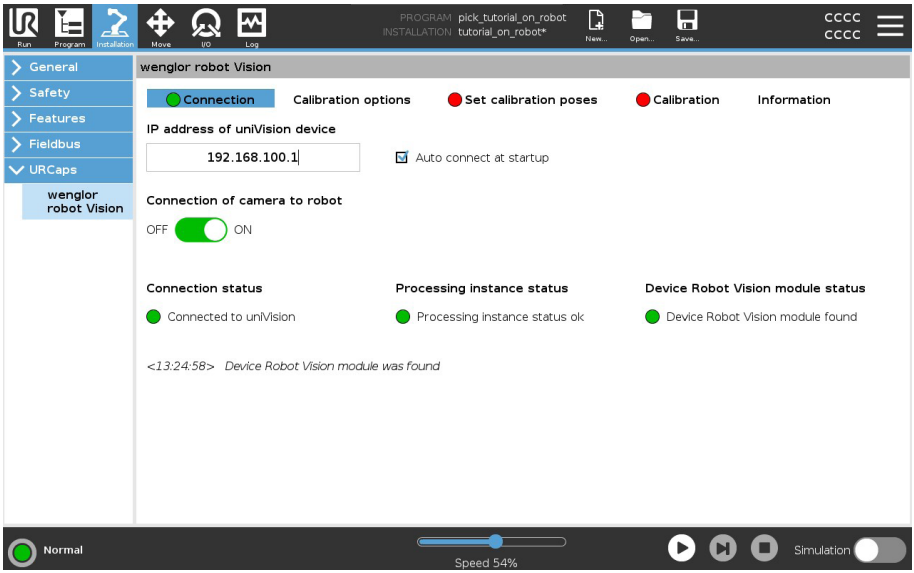
- Connection tab
  - » Red: No connection to Machine Vision Device possible. Check the network setup.
  - » Green: Connection to Machine Vision Device established and valid.
- Set calibration poses tab
  - » Red: Not all calibration poses were set. Check the calibration poses and set them accordingly.
  - » Green: All calibration poses were set.
- Calibration tab
  - » Grey: Could not check for previous calibration data as no connection to the Machine Vision Device is available. Establish the connection to the Machine Vision Device first.
  - » Red: No valid calibration data available. Run the calibration procedure again.
  - » Yellow: Calibration is incomplete (if second calibration step is missing when the camera is not on robot).
  - » Green: Calibration data was set during the current session or was loaded from a previous calibration.



Connection tab

Connect to the Machine Vision Device via entering the IP Address (default 192.168.100.1) and the port (default 6008) and by setting the connection slider to on. By default, auto connect at startup is active.

NOTE!
Make sure that robot and Machine Vision Device are in the same network.



Status:

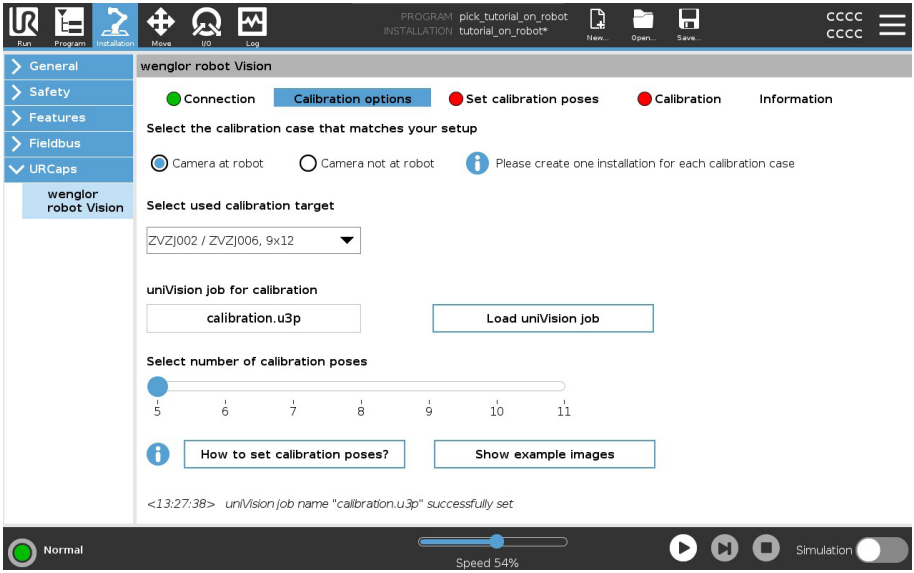
- Connection status
  - » Red: No connection to Machine Vision Device possible. Check the cables and the network setup.
  - » Green: Connection to Machine Vision Device is available.
- Processing Instance status
  - » Red: Processing Instance status displays an error. Check the status of the Processing Instance on the device website of the Machine Vision Device.
  - » Green: No error at status of Processing Instance.
- Device Robot Vision module status
  - » Red: No Device Robot Vision within current uniVision job. Add Device Robot Vision to current uniVision job.
  - » Green: Device Robot Vision is available in current uniVision job.

Calibration options tab

Select if the camera is mounted on the robot or not on the robot.
Select the size of the calibration plate ZVZJ.

Enter the name of the uniVision job for calibration and load it. The job is loaded every time the calibration procedure is started. Set the number of calibration poses between 5 and 11 (default: 5 poses). The calibration results get better with more calibration poses (e.g. seven to eleven poses).

Further info is provided about how to set the calibration poses and some example images from camera perspective for the tab “Set calibration poses”.

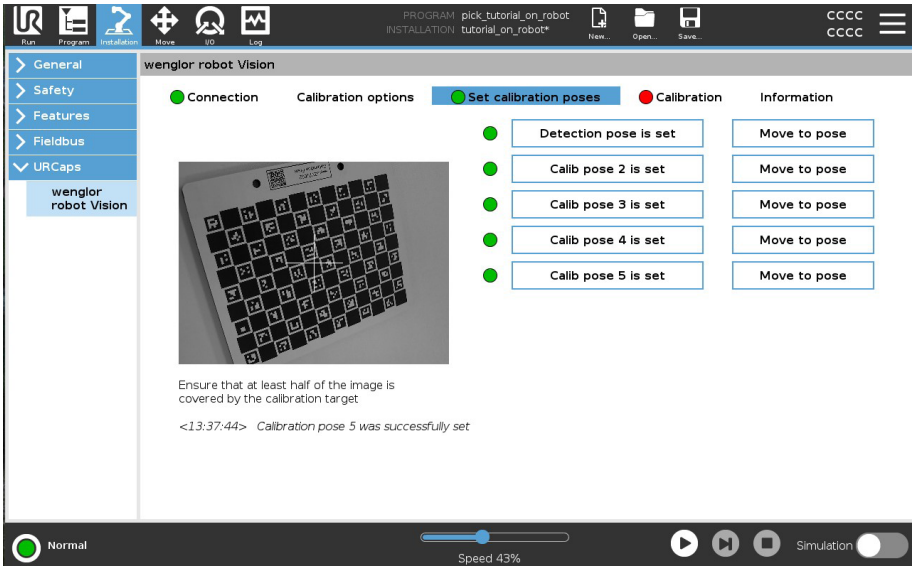


Set calibration poses tab

The calibration poses are different for the camera on robot and for the camera not on robot. The number of buttons to set the calibration poses depends on the settings made in the calibration options tab.

NOTE!

- For details about the calibration poses, see section “5.1 Basics with Robot Server”.
- In case of camera on robot, keep in mind that the first calibration pose is also later the detection pose.



Calibration tab

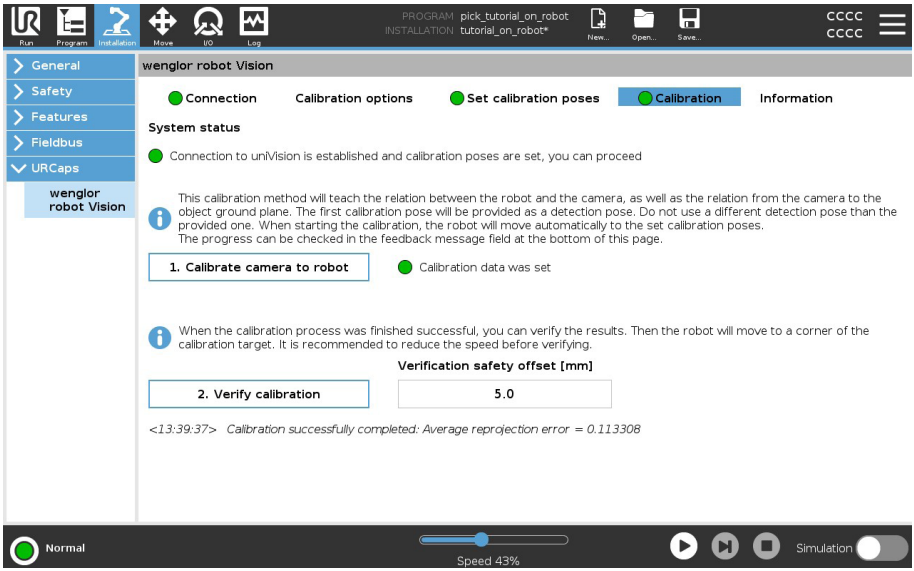
System status LED for calibration:

- Red: No active connection to Machine Vision Device and not all calibration poses set. Connect first to the Machine Vision Device and set the calibration poses.
- Yellow: Only one condition is true
  - » Active connection to Machine Vision Device, but not all calibration poses are set. Set the calibration poses.
  - » No active connection to Machine Vision Device, but the calibration poses are set. Connect first to the Machine Vision Device.
- Green: Active connection to Machine Vision Device and all calibration poses are set.

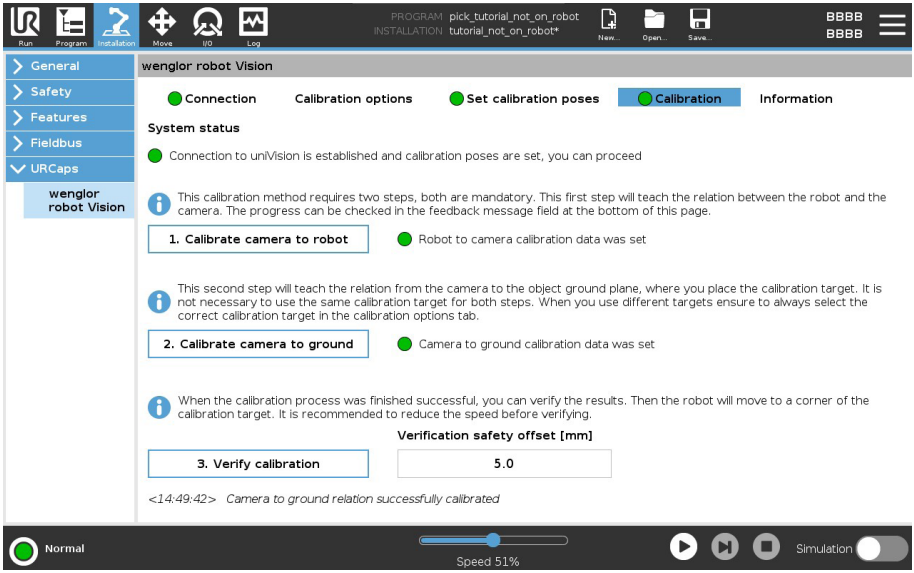
It is only possible to start the calibration if the system status LED is green. At the start of the calibration, the “uniVision job for calibration” is loaded. For each successful calibration, a separate calibration file is created and stored with an ascending number on the Machine Vision Device at /media/card/calibration. By default, the latest calibration file is used. It is possible to load another calibration file on the device website at the tab “Jobs”.

The calibration process differs depending on whether the camera is mounted on the robot or not.

- In case of camera on robot, only one calibration step is needed. When starting the calibration, the robot will load the calibration job and move automatically to the calibration poses set in the tab “Set calibration poses”.



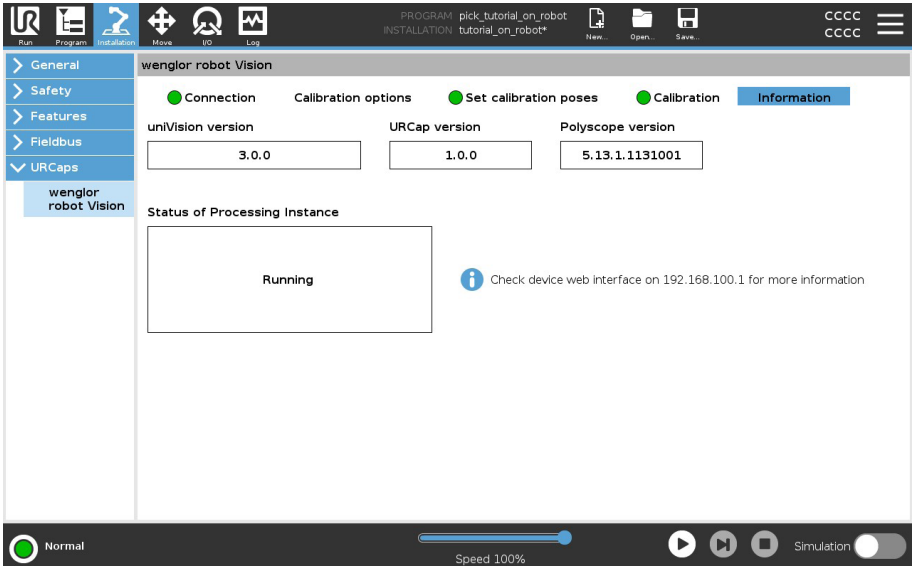
- In case of camera not on robot, the calibration process requires two steps.
  - » When starting the first calibration step, the calibration plate must be mounted at the robot. The robot will load the calibration job and move automatically to the calibration poses set in the tab “Set calibration poses”.
  - » For the second step, unmount the calibration plate from its holder on the robot and place it on the object plane. Make sure that the calibration plate is visible for the camera (robot arm can be moved away in order to not interfere). Check the camera image in the tab “Set calibration poses”. When the calibration plate is placed, you can trigger the second calibration step. Only one image is captured for the second calibration step.



Calculating the calibration results takes some time. Wait until the calibration is completed.

Information tab

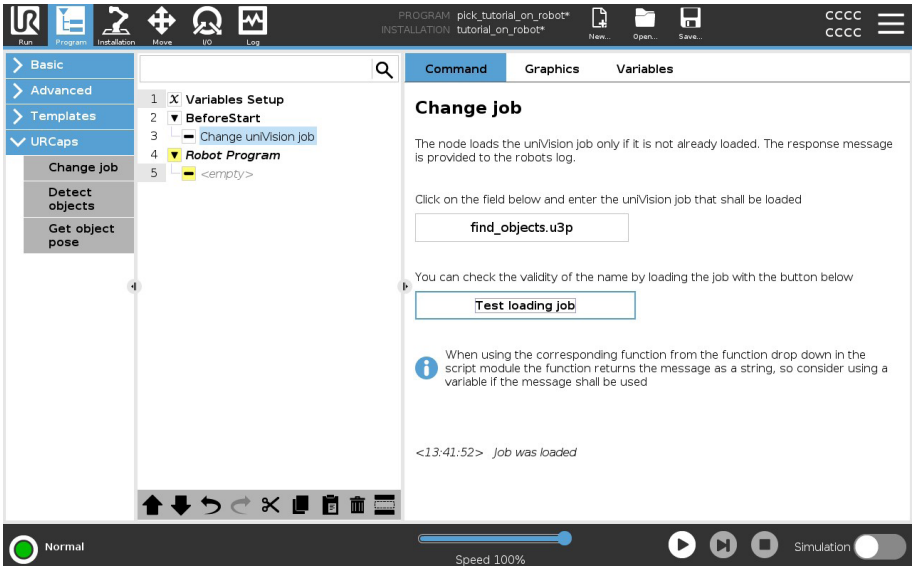
The information tab contains info about different versions as well as the status of the Processing Instance.



5.6.3 UR Program

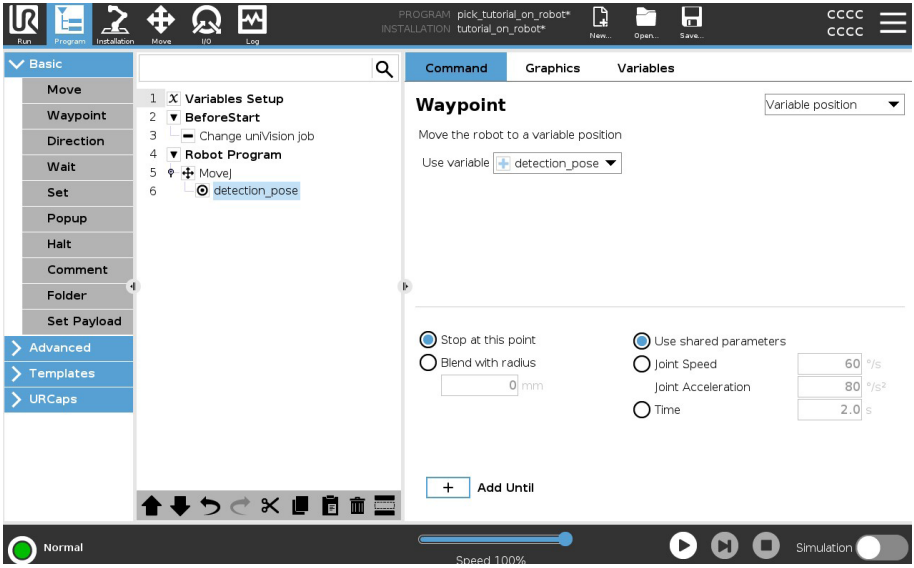
The URCap “wenglor robot vision” adds nodes to change the uniVision job, to detect objects or to get the latest object pose. Furthermore, it is possible to detect the calibration target or to calibrate to the calibration target. Switch to the program tab on UR side in order to create the robot program. First step is to load the uniVision job for detecting objects. Use the “Change job” node from the URCaps section on the left drop-down menu.

Enter the name of the uniVision job file. To test the validity of the job name, load it by clicking the button “Test loading job”.

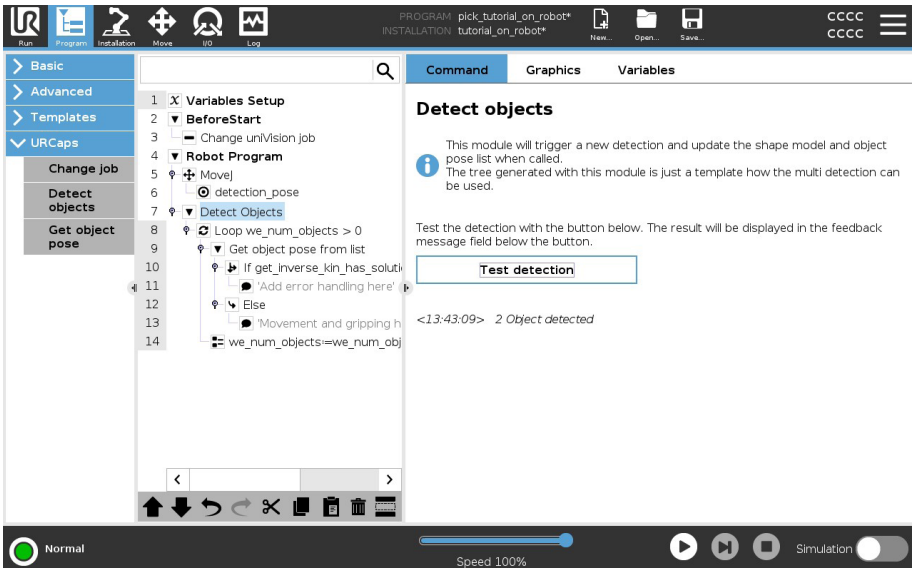


In case of camera not on robot, you can teach a detection pose to prevent the robot from interfering the field of view of the camera while capturing images.

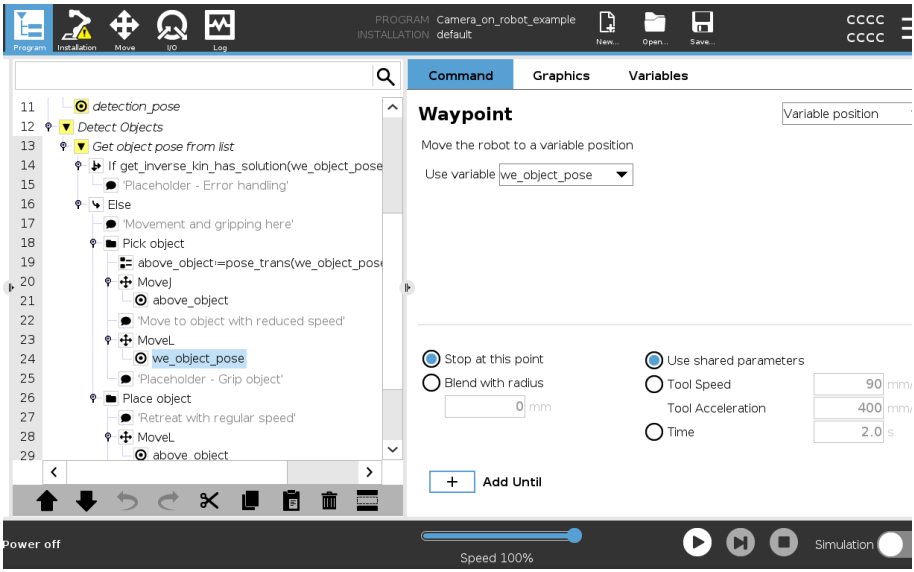
In case of camera on robot, it is necessary to move the robot to the detection pose so that the camera can check for objects. The detection pose from the URCap must be used (identical to the first calibration pose). To update the detection pose, it is necessary to set the detection pose in the URCap installation page and to re-run the calibration process.



Then add the node "Detect objects". It adds a subtree, which iterates through all found objects. In case of finding only one object per image (single object detection), remove the loop by cutting “Get object pose from list”, delete the loop and paste it again.



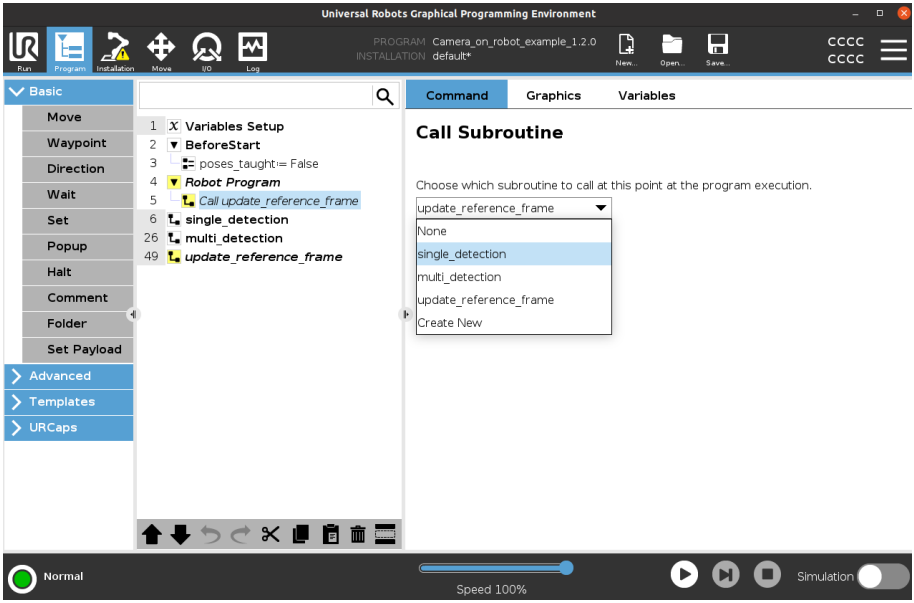
You can access the object pose as a variable in order to move the robot to any position.



The example program contains 3 subroutines:

- single_detection
- multi_detection
- update_machine_reference

Execute the relevant subroutine with the call command.



Single detection:
It loads the detection job, moves to the detection pose and triggers the detection.

```
6  single_detection
7    Change uniVision job
8    MoveJ
9      detection_pose
10   Detect Objects
11     Get object pose from list
12       If we_pose_valid == False
13         'Add error handling here'
14         Popup: Pose not valid.
15         Halt
16       'Movement and gripping here'
17       'Add conditional checks for we_shape_model for we_custom_value'
18       Pick object
19         'Move to object with reduced speed'
20         MoveL
21           we_object_pose
22         'Placeholder - Grip object'
23       Place object
24         'Placeholder - Move to place pose'
25         'Placeholder - Drop object'
```

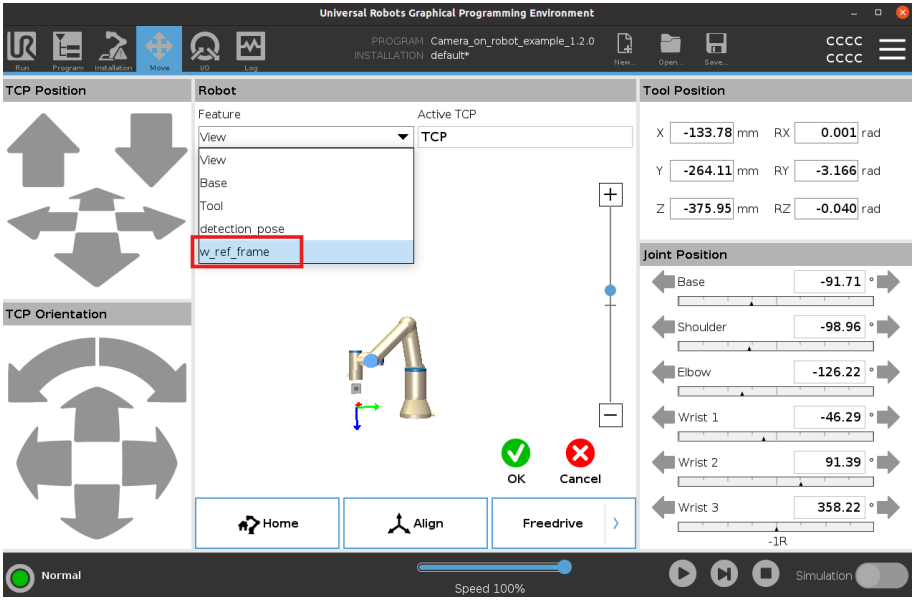
Multi detection:
Similar to single detection. It wraps the single detection in a loop to handle multiple objects within one detection.

```
27  Change uniVision job
28  MoveJ
29    detection_pose
30  Detect Objects
31    Loop we_num_objects > 0
32      Get object pose from list
33        If we_pose_valid == False
34          'Add error handling here'
35          Popup: Pose not valid.
36          Halt
37        'Movement and gripping here'
38        'Add conditional checks for we_shape_model for we_custom_value'
39        Pick object
40          'Move to object with reduced speed'
41          MoveL
42            we_object_pose
43          'Placeholder - Grip object'
44        Place object
45          'Placeholder - Move to place pose'
46          'Placeholder - Drop object'
47        'Necessary to prevent index errors'
48        we_num_objects:=we_num_objects - 1
```

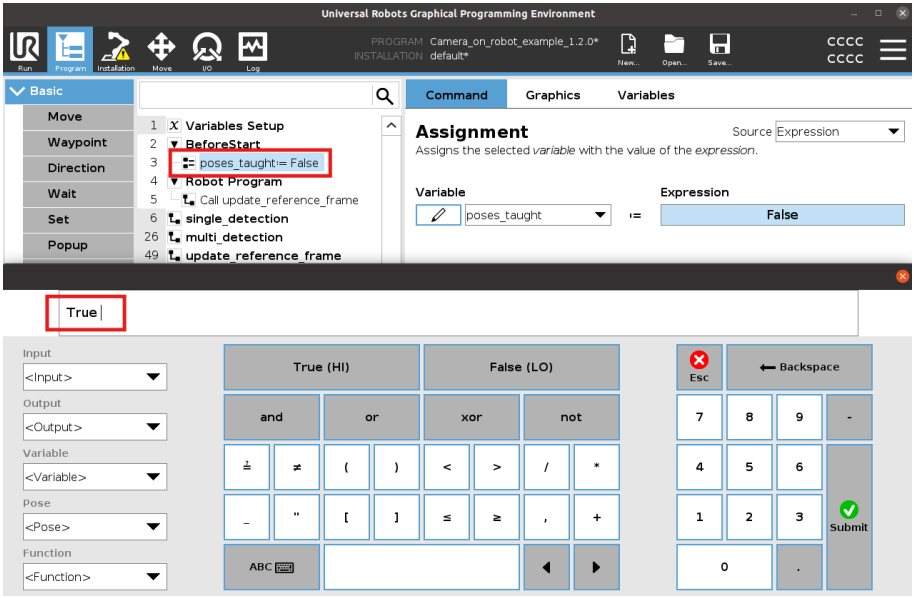
Update machine reference:
It shows how the calibration target pose can be used to update a reference frame and with it also the related poses (see section “5.1 Basics with Robot Server”). At first, you need to teach the dummy pose pose_in_machine with any value. As you teach it again after the update of the reference frame, you can really set any waypoint. The program loads the job and executes the detection. The "Get target pose" node also updates w_ref_frame pose permanently, so the changes remain after a restart of the robot. Remember to save the installation before shutting down the robot as w_ref_frame is saved in the installation, not in the program. But unfortunately, the program reads the installation feature values just at the program start, we also need the assignment node to update it within the program. So the program can also use the updated value.

```
49  update_reference_frame
50    Change uniVision job
51    'Move to pose where target is visible'
52    MoveJ
53      target_detect_p
54    Get target pose
55      If we_pose_valid == False
56        'Add error handling here'
57        Popup: Pose not valid.
58        Halt
59    w_ref_frame:=we_target_pose
60    If poses_taught == False
61      Popup: 'Reference frame updated. Please update poses now relative to w_ref_frame'
62      Popup: 'Set poses_taught to True and restart the program after teaching the poses.
63      Halt
64    'The machine reference frame is now updated, and so are the poses inside the machine.'
65    'Move to poses in the machine that were taught relative to `w_reference_frame`.'
66    MoveL
67      pose_in_machine
```

After the w_ref_frame update, you can teach your poses to the updated w_ref_frame. It is important to select the w_ref_frame as the feature.



After teaching the poses, set the variable poses_taught to True (in the "Before start" section at the top of the robot program).



Then, you can run the program and the reference will be updated before the robot moves to the updated related dummy poses. In this case just pose_in_machine.

NOTE!
The robot program example is available on https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

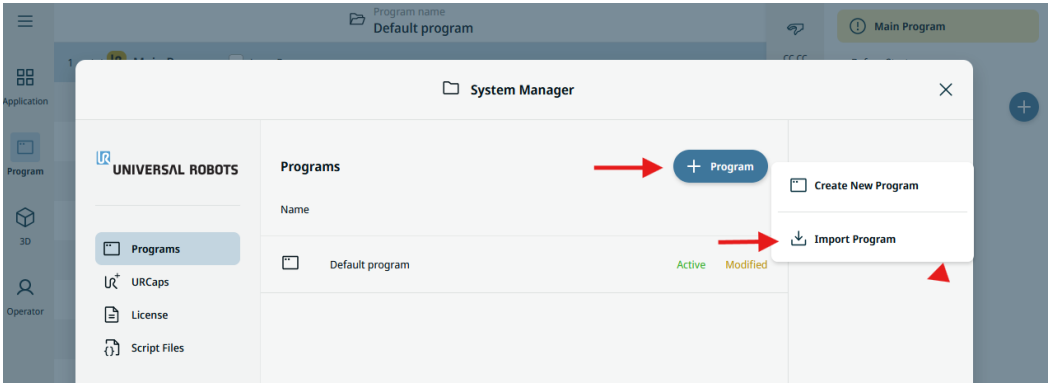
5.7 UR Robots with Polyscope X Script

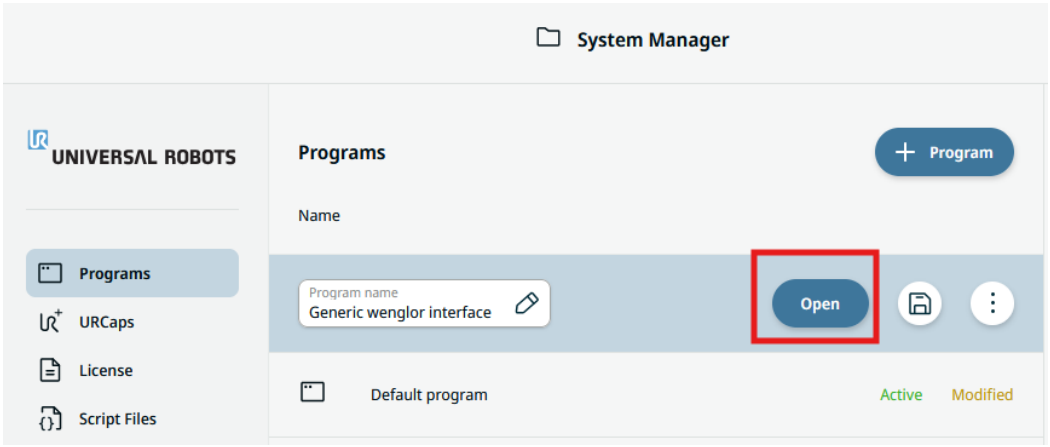
Supported by UR robots with Polyscope X.

The robot example consists of the following file:

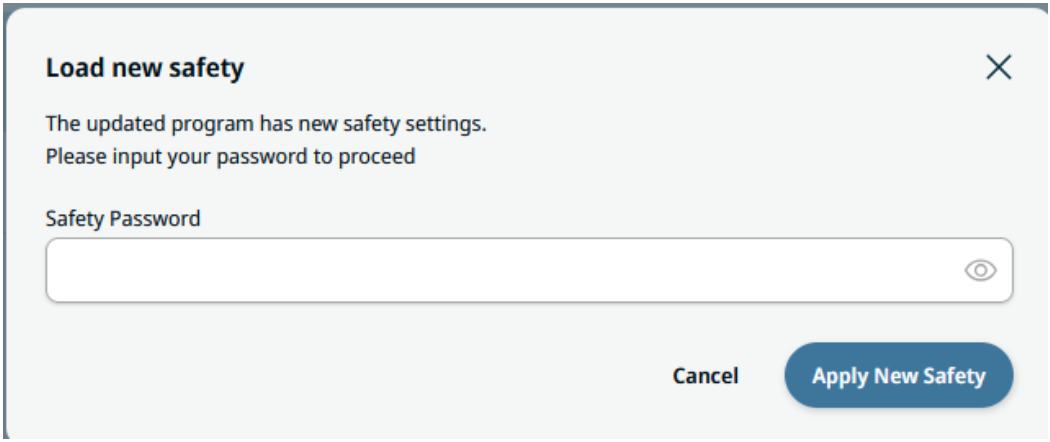
- Generic wenglor interface.urpx

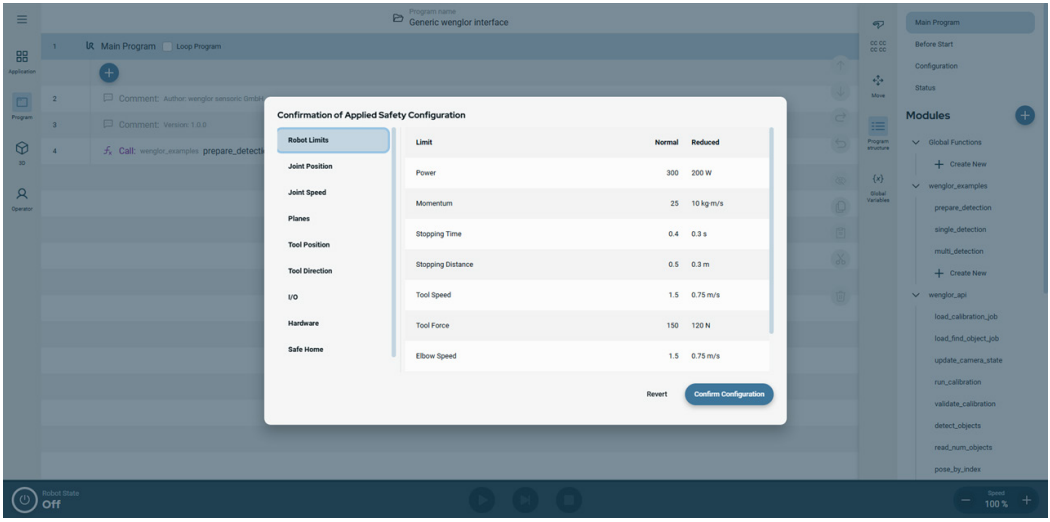
Import and open the file "Generic wenglor interface.urpx" on the UR robot.





Enter the passwort of the UR robot (by default "ursafe").





Adjust the user configuration.

Connection:

- WG_CAM_IP: IP address of Machine Vision Device (by default 192.168.100.1)
- WG_CAM_PORT: TCP port of the robot vision server (by default 32006)
- WIG_COMMAND_TIMEOUT: Socket read timeout in seconds (by default 20)

uniVision jobs:
Configure three uniVision jobs. The suffix ".u3p" is mandatory and the name must not be wrapped in quotes.

- WG_CALIB_JOB: Job for calibration. Used by wenglor_api.run_calibration (by default calibration.u3p)
- WG_FIND_OBJECTS_JOB: Job used by single_detection and multi_detection (by default find_objects.u3p)
- WG_FIND_TARGET_JOB: Job used by update_reference_frame (by default find_target.u3p)

Use case and calibration plate:

- WG_USE_CASE: Either camera_on_robot or camera_not_on_robot (default camera_on_robot).
- WG_CALIBRATION_TARGET: Defines the calibration plate (by default zvzj001). Select zvzj001, if using ZVZJ005 and select zvzj002, if using ZVZJ006).

Poses:

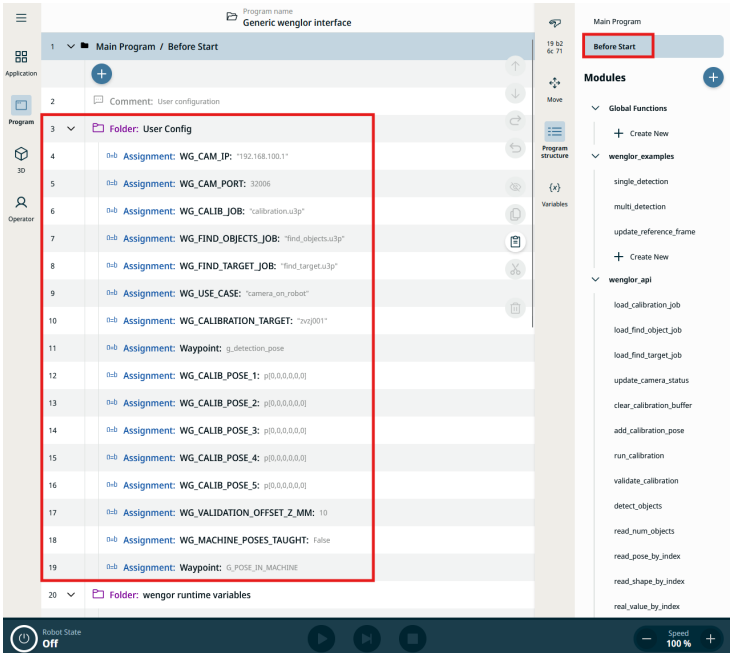
- g_detection_pose: Pose the robot moves to for object detection and validation. Teach the pose.
- WG_CALIB_POSE_1 … WG_CALIB_POSE_5: The five calibration poses. Teach each pose (the template provides exactly five slots).

Validation:

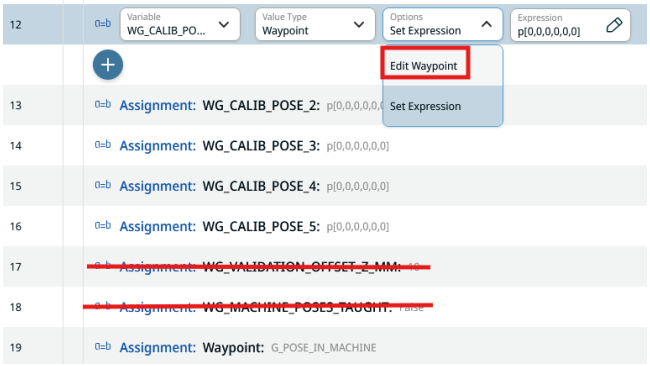
- WG_VALIDATION_OFFSET_Z_MM: Retract distance along the tool Z axis (mm) when the robot approaches the detected calibration target pose during validate_calibration. Provides a safety gap for visual inspection (default 10).

Reference frame workflow (update_reference_frame only)

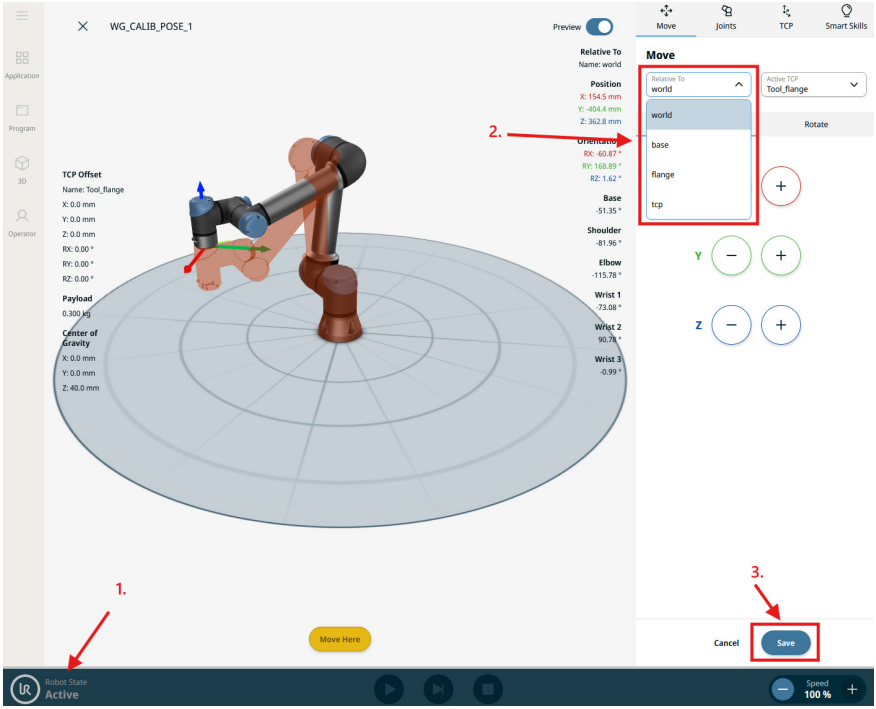
- WG_MACHINE_POSES_TAUGHT: Set to True after teaching poses relative to w_ref_frame (by default False).
- G_POSE_IN_MACHINE: Pose expressed in w_ref_frame that the robot should move to once the machine reference frame has been updated (by default taught).



The calibration poses and the dummy pose G_POSE_IN_MACHINE need to be taught. To do it, select “Set Expression” and change it against “Edit Waypoint”. Then you can teach the poses afterwards.

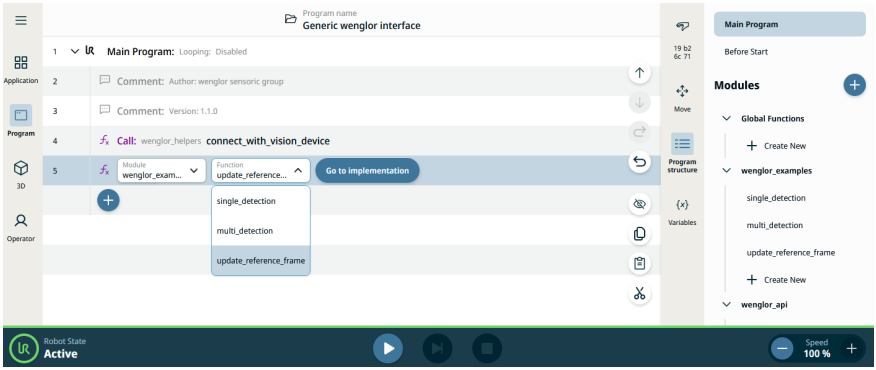


For G_POSE_IN_MACHINE you have to keep in mind to set the reference frame to w_ref_frame. It will appear after update_machine_reference ran at least once.
1. Robot needs to be active to move it while teaching the pose.
2. Select the reference frame, for G_POSE_IN_MACHINE set w_ref_frame, otherwise world.
3. Save the pose.



Main program

All examples live in the wenglor_example module



single_detection
1. Moves to g_detection_pose
2. Loads WG_FIND_OBJECTS_JOB.
3. Sends detect[…] and moves the robot to the returned object pose.
4. Reads shape model and additional value of the detected object (wg_shape_model, wg_additional_value).

multi_detection
Same as single_detection, but iterates through every object returned by num_objects:get and reads pose, shape and additional value for each.

update_reference_frame (see section “5.1 Basics with Robot Server”)
1. Loads WG_FIND_TARGET_JOB, moves to g_detection_pose and triggers a calibration target detection.
2. Creates (or updates) the frame w_ref_frame attached to world.
3. On the first run (WG_MACHINE_POSES_TAUGHT = False), the program stops so the operator can teach G_POSE_IN_MACHINE relative to w_ref_frame. Set WG_MACHINE_POSES_TAUGHT = True and restart the program.
4. On subsequent runs, the robot moves to G_POSE_IN_MACHINE.

This workflow lets you re-localize a machine or fixture automatically between runs without re-teaching downstream poses.

Validation
Validation can be carried out once a calibration is present on the vision device (e.g. the second bit of state[WG_USE_CASE] is 1).

- In the Main Program, add a call to Module wenglor_api → function validate_calibration (script function wenglor_api_validate_calibration).

What validate_calibration does:
1. Checks the camera / calibration state via state[WG_USE_CASE].
2. Prompts the operator and moves to g_detection_pose.
3. Sends validate[WG_USE_CASE, current_tcp_pose] to the vision device.
4. Moves the robot to the returned calibration-target pose, offset upwards by WG_VALIDATION_OFFSET_Z_MM along the tool Z axis, so the operator can visually confirm the result.

Prerequisite: The calibration plate must be visible from g_detection_pose.



NOTE!
The robot program example is available on https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

5.8 KUKA Robots

The example was tested with the KUKA KRC4 robot controller, KR 6 R1820 Arc Robot Arm, software KSS 8.3.33 with EthernetKRL 3.2.4. Additionally, it was tested with KRC5 robot controllers.

NOTE!
The robot program example is available on https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

The KUKA robot vision example consists of the following files:

- wenglorGlobal.dat
- wenglorGlobal.src
- wenglorMain.dat
- wenglorMain.src
- wenglorUserConfig.dat
- wenglorUserConfig.src
- wenglorVision.xml

Edit the file "wenglorVision.xml"

- Adjust the IP address of the Machine Vision Device (by default: 192.168.100.1).
- Adjust the port of the robot server (by default: 6008).

[KRC4(WINDOWS-BIDO66S)] - KRC:\R1\Program\KUKA_wenglorRobotVision_2.0.0\wenglorVision.xml

```
<ETHERNETKRL>
   <CONFIGURATION>
      <EXTERNAL>
         <IP>192.168.100.1</IP>          <-- IP address of device
         <PORT>6008</PORT>               <-- Port of robot server
         <TYPE>Server</TYPE>
      </EXTERNAL>
      <INTERNAL>
         <ENVIRONMENT>Program</ENVIRONMENT>
         <BUFFERING Mode="FIFO"/>
         <TIMEOUT Connect="60000"/>
         <ALIVE Set_Flag="1" Ping="200"/>
         <PROTOCOL>TCP</PROTOCOL>
         <MESSAGES Logging="error" Display="error"/>
      </INTERNAL>
   </CONFIGURATION>
   <RECEIVE>
      <XML>
         <ELEMENT Tag="sensor/reply/command" Type="STRING"/>
         <ELEMENT Tag="sensor/reply/return_code" Type="INT"/>
         <ELEMENT Tag="sensor/reply/job_name" Type="STRING"/>
         <ELEMENT Tag="sensor/reply/state" Type="STRING"/>
         <ELEMENT Tag="sensor/reply/calib_accuracy" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/num_objects" Type="INT"/>
         <ELEMENT Tag="sensor/reply/object_pose/@X" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/object_pose/@Y" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/object_pose/@Z" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/object_pose/@RX" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/object_pose/@RY" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/object_pose/@RZ" Type="REAL"/>
         <ELEMENT Tag="sensor/reply/shape_model" Type="INT"/>
         <ELEMENT Tag="sensor/reply/additional_value" Type="STRING"/>
         <ELEMENT Tag="sensor/reply" Set_Flag="42"/>
      </XML>
   </RECEIVE>
   <SEND/>
</ETHERNETKRL>
```

Copy the file “wenglorVision.xml” on the robot controller in the file location:
C:\KRC\ROBOTER\Config\User\Common\EthernetKRL\

Edit the file "wenglorUserConfig.src":

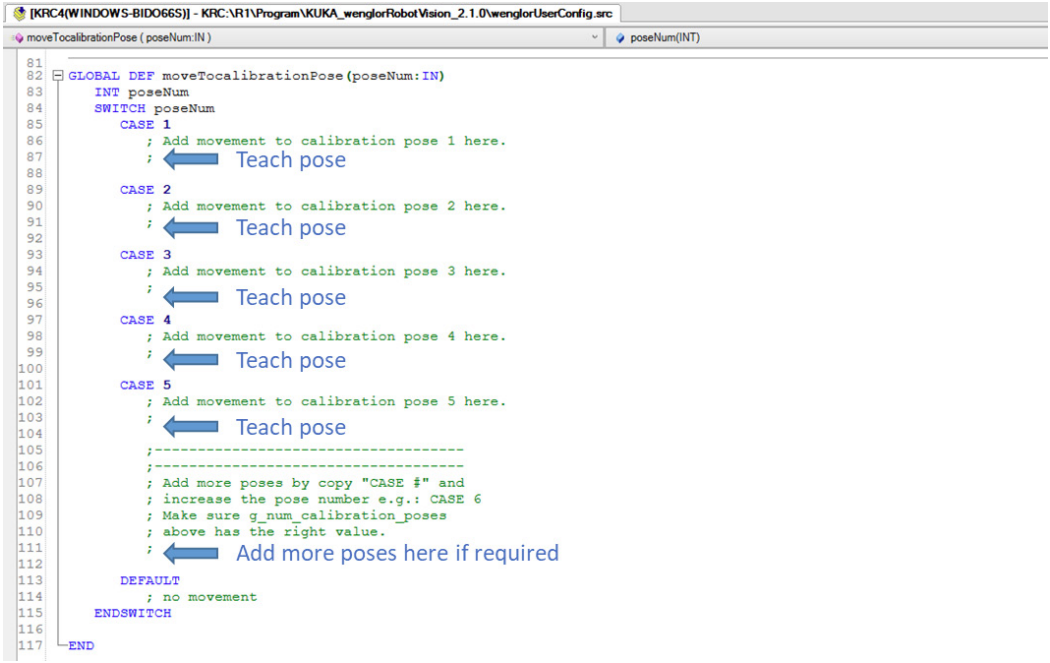
- The name “wenglorVision” at “W_CONNECTION[ ]” must match to the name of the XML file. By default, it fits automatically (do not rename).
- Select the use case “camera_on_robot” or “camera_not_on_robot”.
- Select the size of the calibration plate. Select ZVZJ001 if using ZVZJ005 and select ZVZJ002 if using ZVZJ006.
- Enter the number of calibration poses (by default 5).
- Enter the uniVision job names
- Enter the safety offset for the validation
- Select the user command (singleDetection, multiDetection or updateReferenceFrame)

For mobile platform use case (see section “5.1 Basics with Robot Server”):

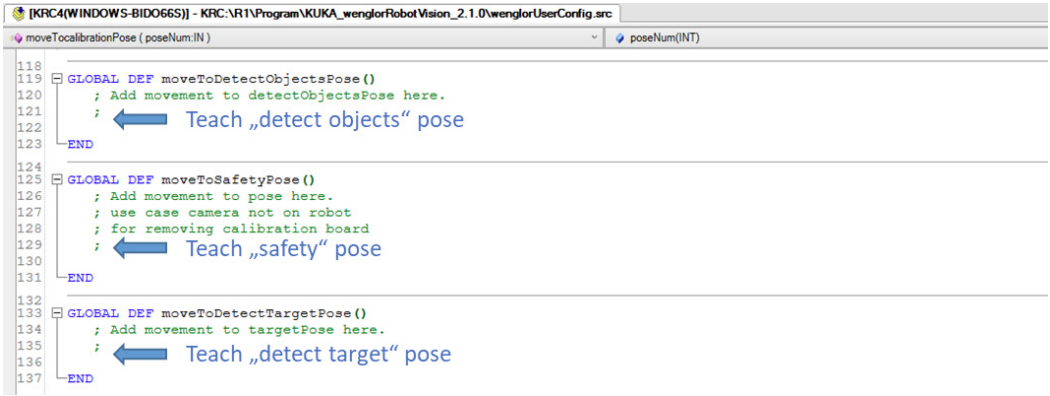
- Enter base number and name in case of update reference frame
- Set "W_MACHINE_POSES_TAUGHT = TRUE" after teaching poses relative to the reference frame



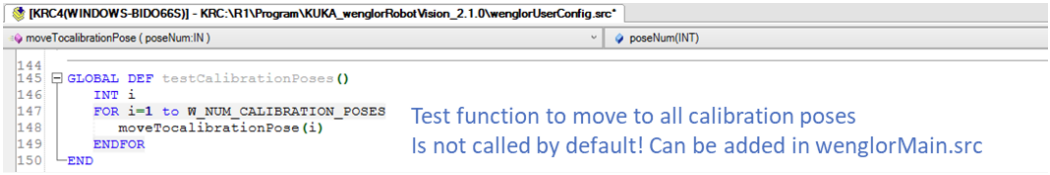
- Teach the calibration poses (minimum five poses).
- More poses can be added if needed. The number of robot poses must be equal to the number of “W_NUM_CALIBRATION_POSES”.



- Teach the detection pose.
- Teach the safety pose (For the use case “camera not on robot” to move to a pose in order to remove the calibration plate).
- Teach the detect target pose in case of update reference frame.



Test function to move to all calibration poses. Must to be called in wenglorMain.src.



Run "wenglorMain.src" to start the calibration and the detection.

NOTE!
For details about the communication to the robot server, see section “5.13 Generic Robot Vision API”

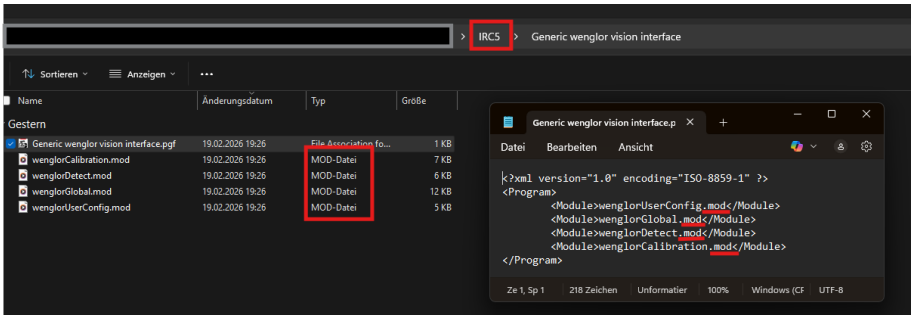
5.9 ABB Robots

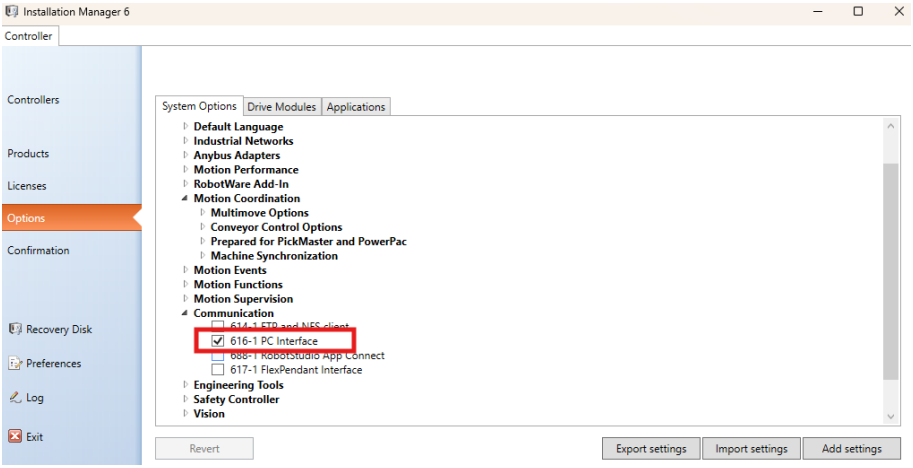
The robot vision example for ABB consists of the following files:

- Generic wenglor vision interface.pgf
- wenglorCalibration.modx
- wenglorDetect.modx
- wenglorGlobal.modx
- wenglorUserConfig.modx

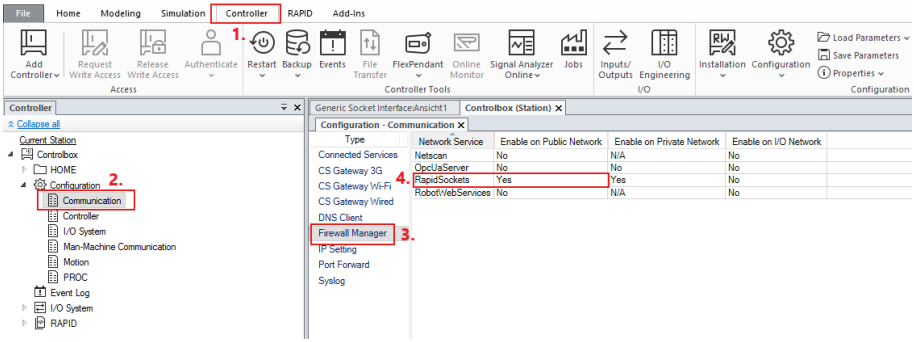
NOTE!

The robot example is available on https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision. It works for OmniCore robot controllers and requires the minimum software version RobotWare 7.3.2. Rename the module files from modx to mod, change the modx files to mod within the PGF file and activate the Option PC Interface in order to run the examples on IRC5 robot controllers (requires minimum software version RobotWare 5.15).

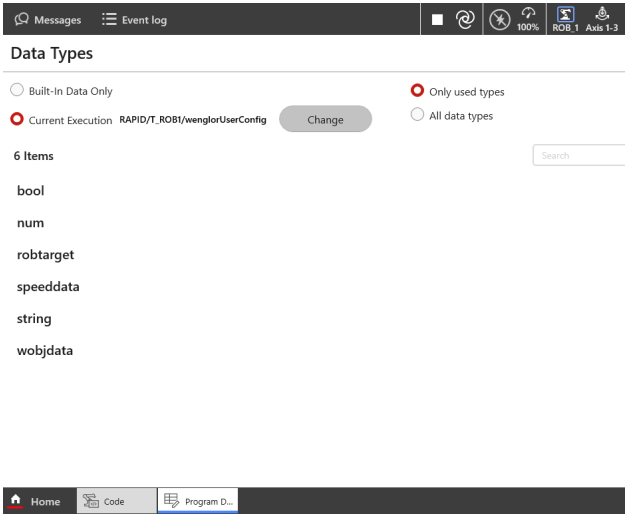




In the ABB software RobotStudio adjust the network settings of the ABB robot controller for the public network and make sure that RapidSockets is set to YES (Communication → Firewall Manager).

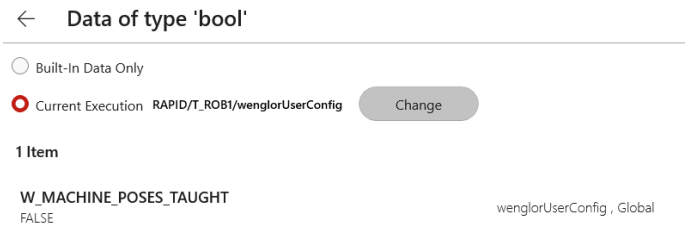


Adjust the parameters in wenglorUserConfig.modx according to your needs:



Boolean parameters (bool)

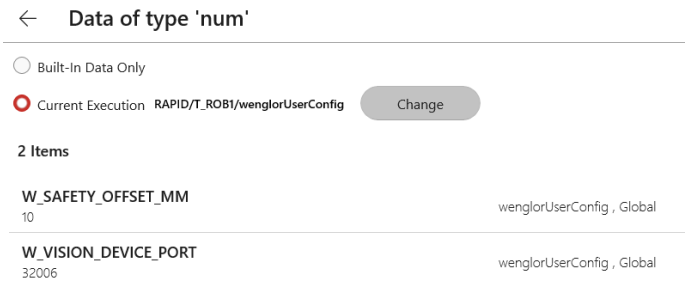
W_MACHINE_POSES_TAUGHT: Used within the updateReferenceFrame use case to stop the program, so the related poses for the reference frame can be taught after the initial reference frame was set. After teaching the poses, set this value to TRUE.



Numerical parameters (num)

W_SAFETY_OFFSET_MM: Specifies the Z offset in millimeter for the validation of the calibration.

W_VISION_DEVICE_PORT: Specifies the communication port for the Machine Vision Device (by default 32006).

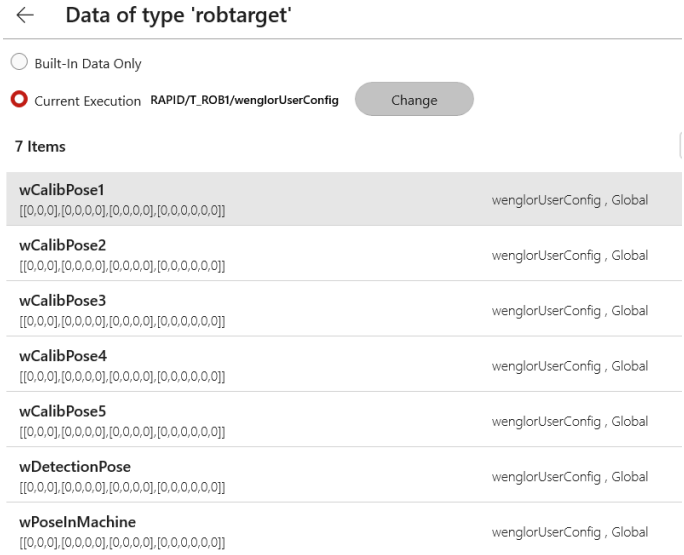


Position data(robtarget)

wCalibPose1 to wCalibPose5: Defines the calibration poses. For details about the calibration poses, see section “5.1 Basics with Robot Server”.

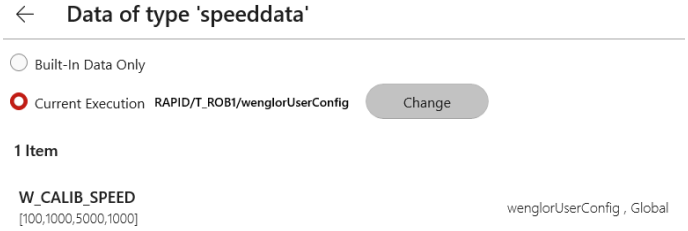
wDetectionPose: Defines the detection pose for the object recognition. Keep in mind that in case of camera on robot, the detection pose needs to be identical to the first calibration pose (handled automatically within the robot example program). In case of camera not on robot, the detection pose must be set separately so that the robot arm is not interfering with the camera image. The detection pose is used for the second calibration step and for the detection step.

wPoseInMachine: Dummy pose that is used to show the use case for updateReferenceFrame and how to use your poses there. This pose is set relative to wReferenceFrame.



Speed data (speeddata)

W_CALIB_SPEED: Defines the speed settings during calibration (in mm/s).



String parameters (string)

W_VISION_DEVICE_IP: Defines the IP address of the Machine Vision Device (by default 192.168.100.1).

W_CALIBRATION_JOB: Defines the name of the uniVision job for calibration.

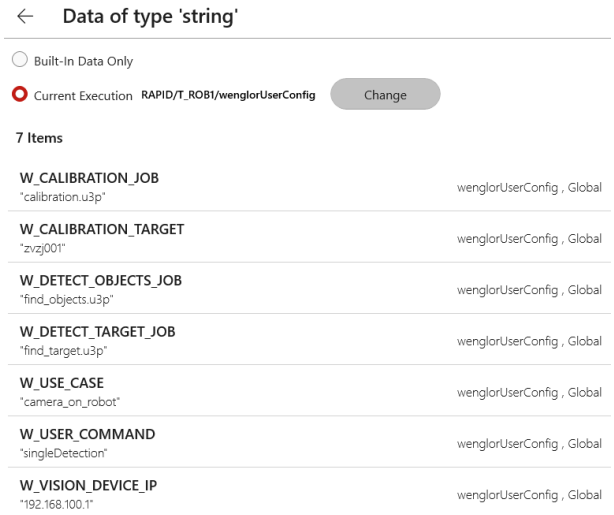
W_CALIBRATION_TARGET: Defines the size of the calibration plate. Select ZVZJ001 if using ZVZJ005 and select ZVZJ002 if using ZVZJ006.

W_DETECT_OBJECTS_JOB: Defines the name of the uniVision job for detection.

W_USE_CASE: Defines if the camera is on the robot or not (e.g. camera_on_robot or camera_not_on_robot).

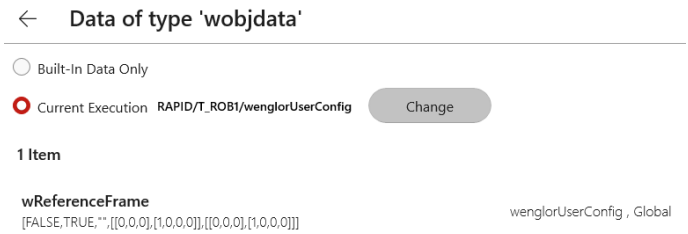
W_DETECT_TARGET_JOB: Defines the name of the uniVision job for detecting the calibration plate.

W_USER_COMMAND: Defines the routine to execute (singleDetection, multiDetection or updateReference-Frame).



Reference frame parameters (wobjdata)

wReferenceFrame: The reference frame that is updated in the routine updateReferenceFrame and where all the poses in the machine shall be taught to, so they can be updated together with the wReferenceFrame. For details, see section “5.1 Basics with Robot Server”.



Loading the program

If you updated the user configuration to your setup, you can load the program "Generic wenglor vision interface". The program is now ready to be executed.



NOTE!
For details about the communication to the robot server, see section "5.13 Generic Robot Vision API".

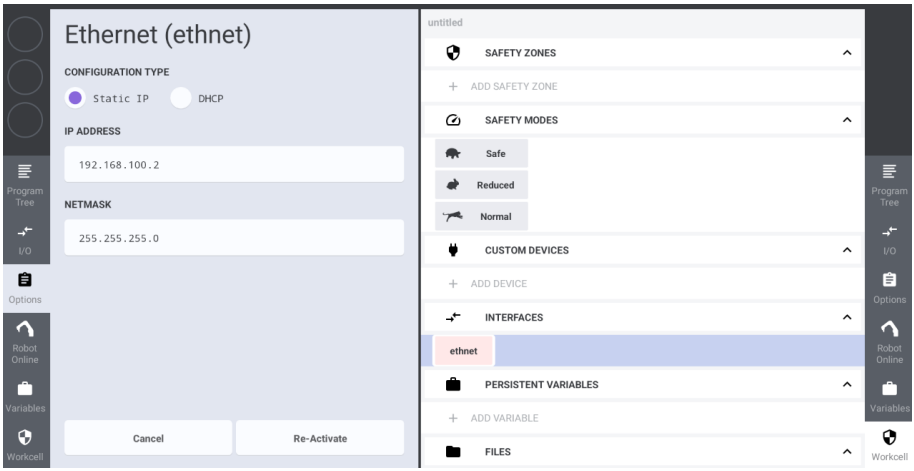
5.10 Kassow Robots

The example was tested with the Robot Controller AC PROFINET, the robot arm KR0810 and the software version Firefly 4.1. CBUN and robot example program are compatible with KR Series and Edge Edition and firmware FireFly.

NOTE!
The robot example program is available on www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

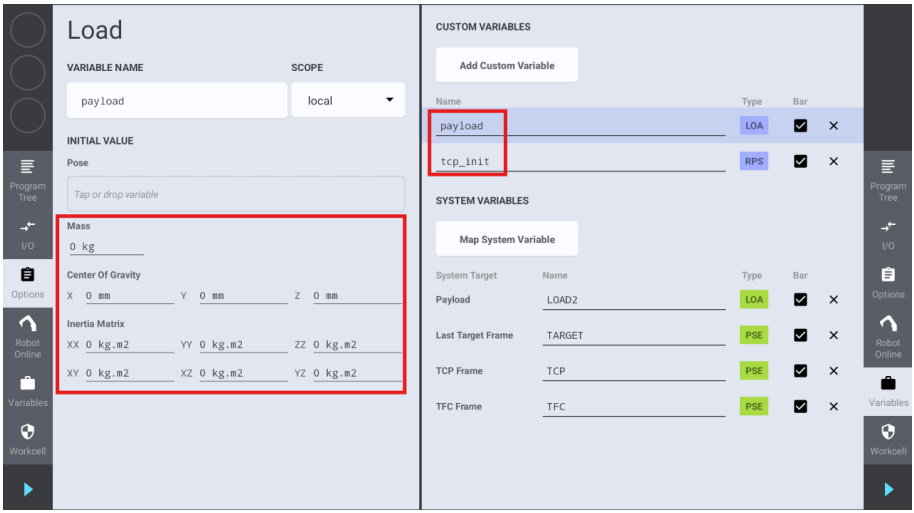
5.10.1 Network Configuration of Robot Controller

To set the IP address of the robot controller, go to Workcell → Interfaces → ethnet. For example, use the IP address 192.168.100.2 and the subnet mask 255.255.255.0 at the robot controller.



5.10.2 Set Payload and TCP

To set the payload and the TCP for the tool, go to Variables and select payload or TCP.



5.10.3 Installation of CBUNs

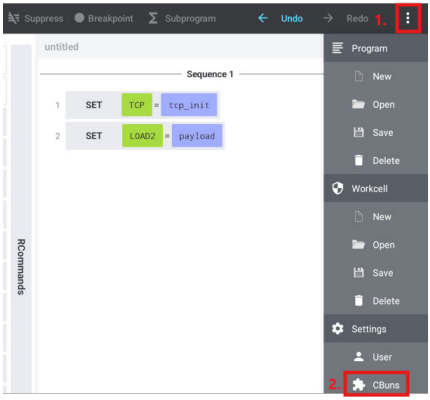
Install the relevant CBUNs in order to use the robot example program.

- System Utils for Program Control
- wenglor vision devices

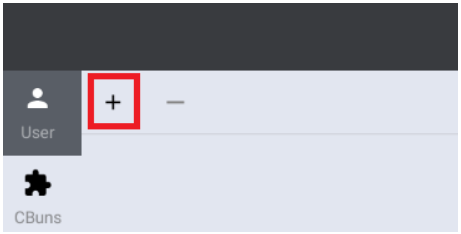
System Utils for Program Control

In the robot example program, the System Util CBUN is required in order to exit the program execution if some conditions are not met.

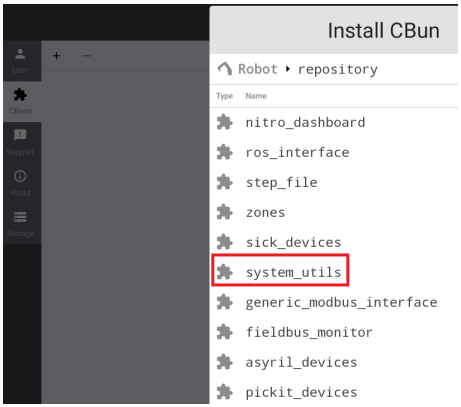
Navigate to the CBUN view to add or remove CBUNs.



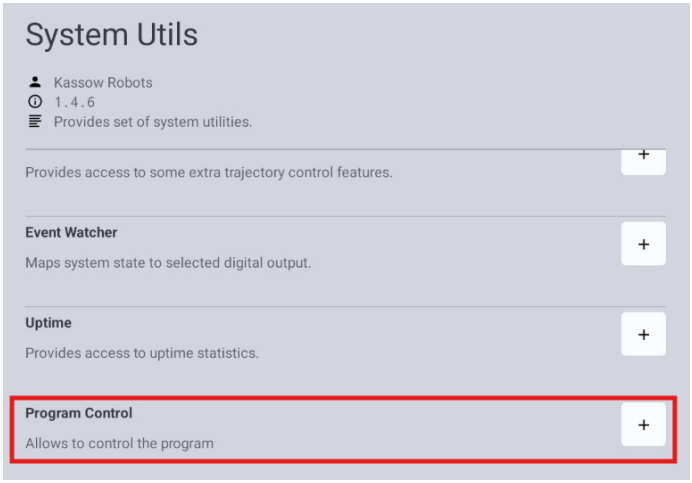
Click on the + icon to install a CBUN.



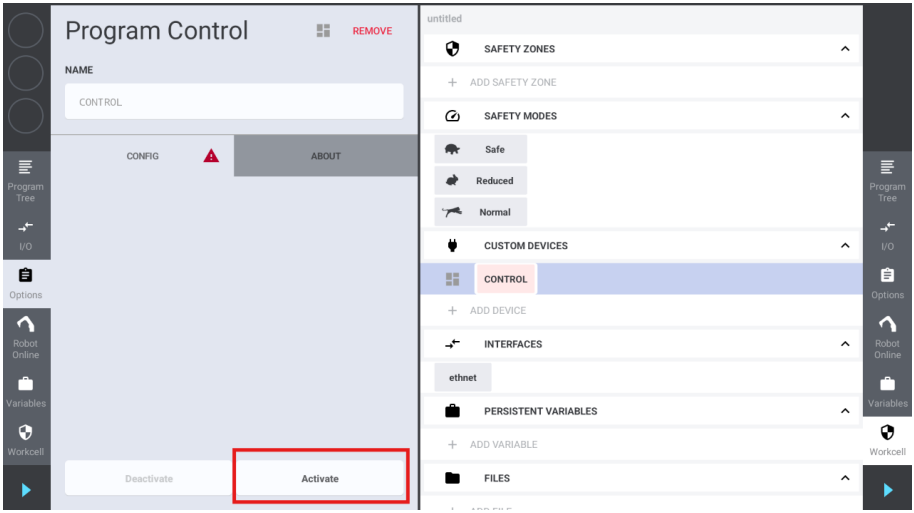
Install system_utils. It should be available on the robot controller. Otherwise, contact your Kassow Robots partner.



Add the method Program Control from the System Utils CBUN.



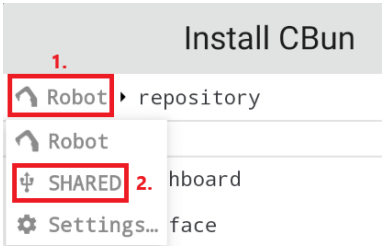
Activate the Program Control Custom Device with its default configuration.



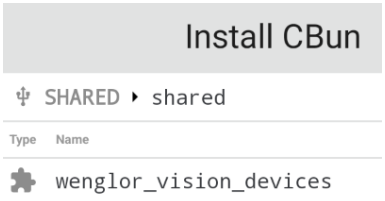
wenglor CBUN

Download the latest wenglor CBUN version from https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision and put the file on a freshly formatted stick.

Plug in the USB stick in the robot controller. Click on Robot and then on SHARED to install the wenglor CBUN from the USB stick.



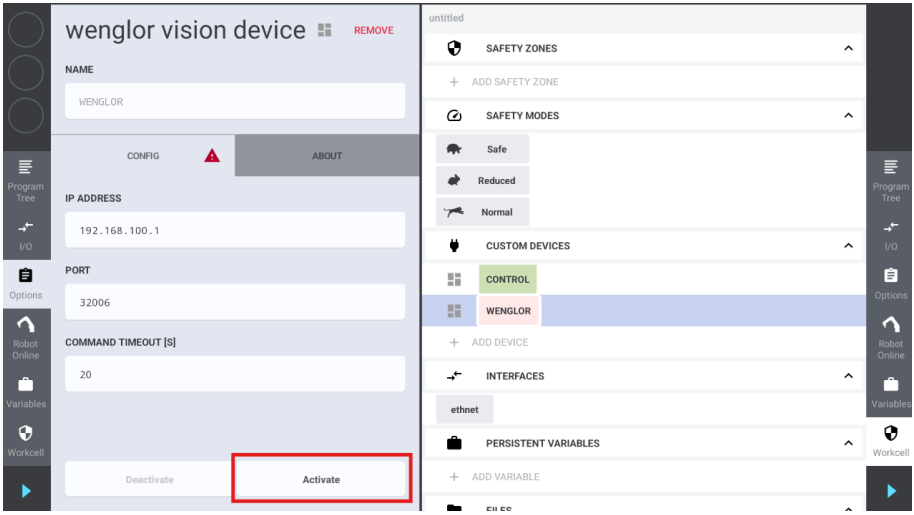
Select wenglor_vision_devices to install it.



Click on the + icon to add the wenglor vision device.



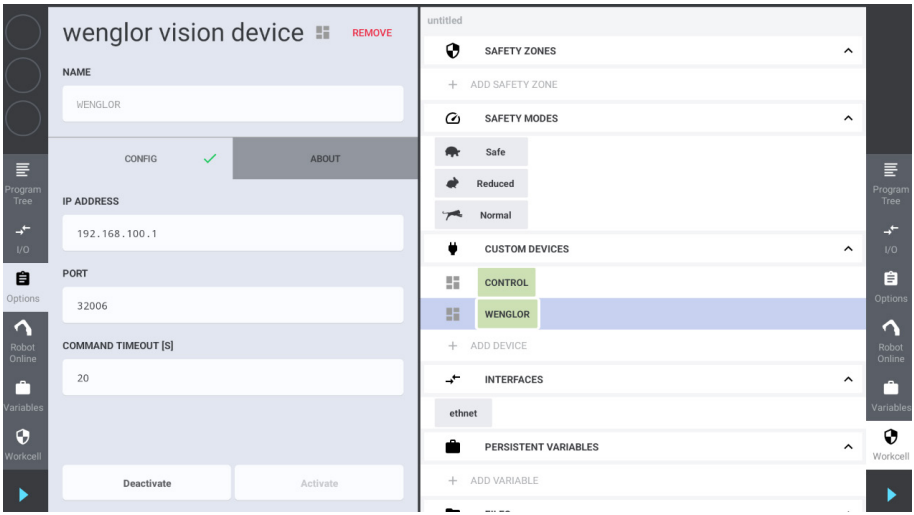
Activate the wenglor vision device in order to initiate a connection from the robot controller to the vision device.



In case of a valid network configuration to the Machine Vision Device, the device is activated successfully.

NOTE!

- The connection is only maintained during program execution. So the Workcell state might become outdated.
- Check the status for the robot connection on the device website of the Machine Vision Device (Tab “Jobs” → Robot Server, see section “5.2 Settings on Device Website”).

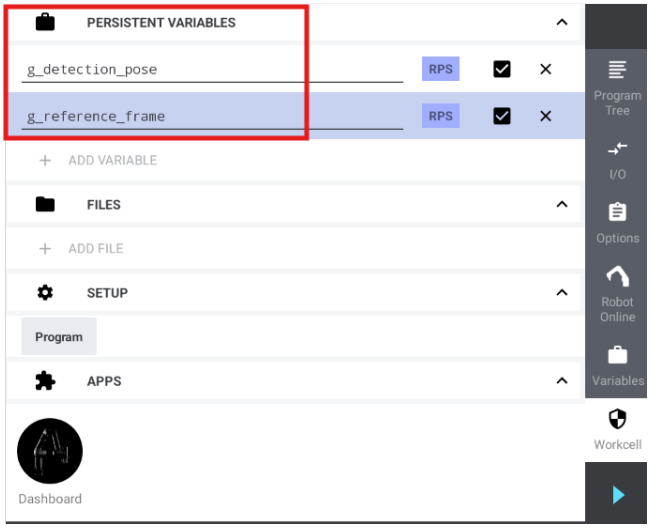


5.10.4 Set up Workcell

For the program execution, two persistent and global variables are required. Add them manually to the workspace.

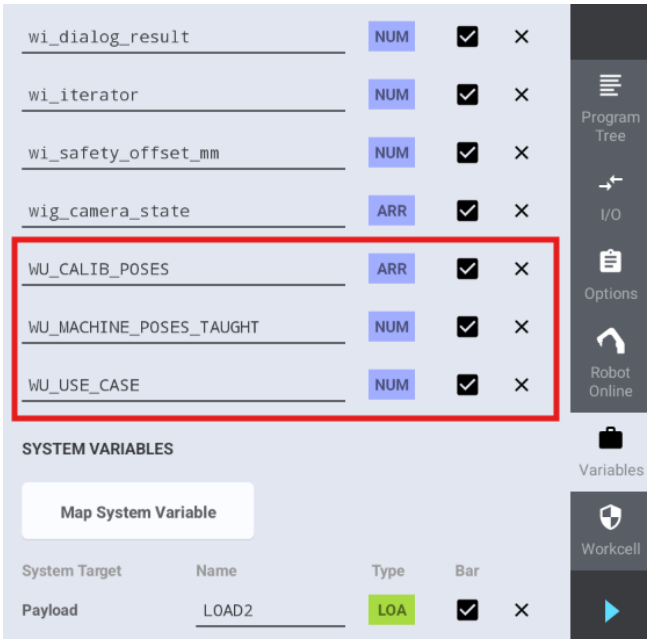
Variable name	Function
g_detection_pose	The pose that triggers the detection. Also used as a retreat pose for calibration step 2 if the camera is not mounted on the robot. The robot will move there, so you can remove the calibration plate to place it on the object ground.
g_reference_frame	The pose that is used as a reference frame for other poses. It is updated in the program to also update the related poses.

Add the variables in the Workcell as persistent variables.



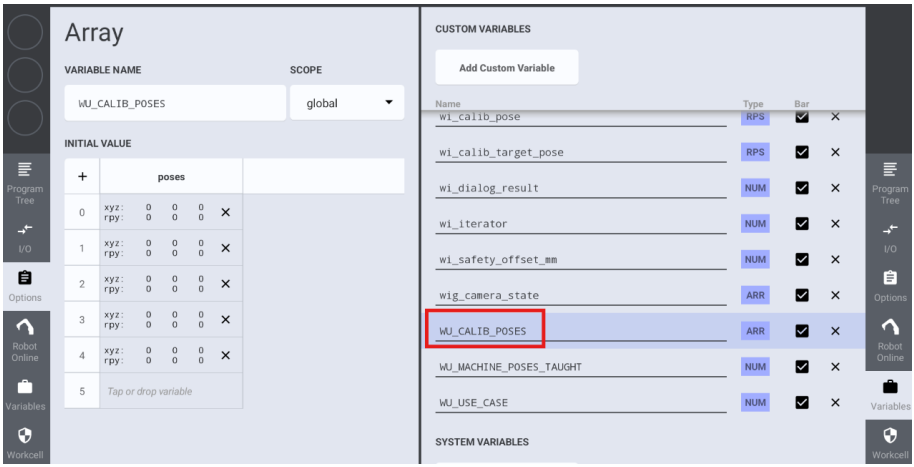
5.10.5 Open Program and Setup Variables

Open the program from the USB stick to set up the variables.

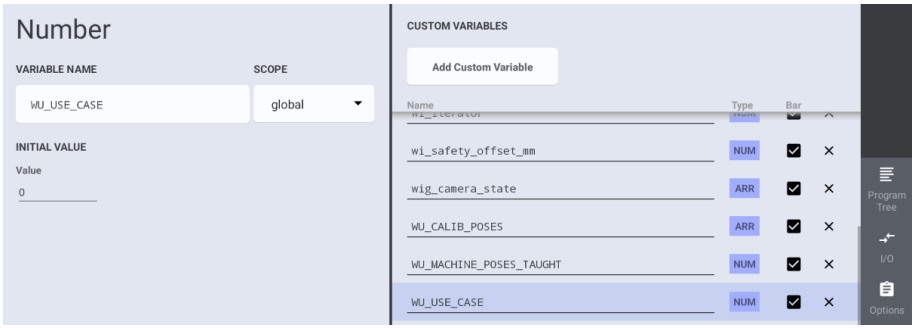


- WU_CALIB_POSES: Is an array of the calibration poses. During the calibration procedure, the program iterates through the poses. You can add/remove/update poses here. A minimum number of five poses is required.
- WU_MACHINE_POSES_TAUGHT: Is a numeric flag that is used to separate the update_reference_frame use case into the preparation and the runtime. Before teaching the poses related to the reference frame, the reference frame needs to be set. When this is done, you can set the num value to 1.
- WU_USE_CASE: Is a numeric flag to indicate where the camera is mounted. Set the value 0, if the camera is mounted on the robot. Set the value 1, if the camera is not mounted on the robot.

Add, remove or update the pose variables.



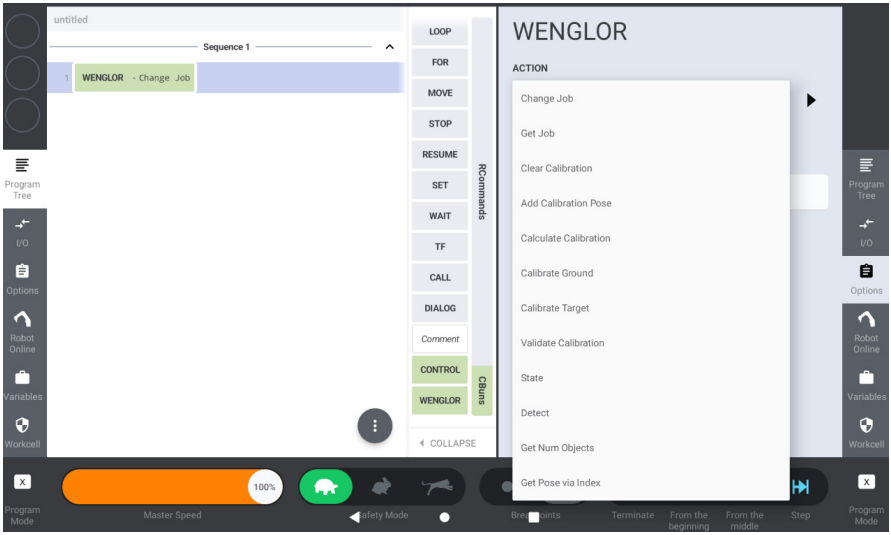
Select the use case.



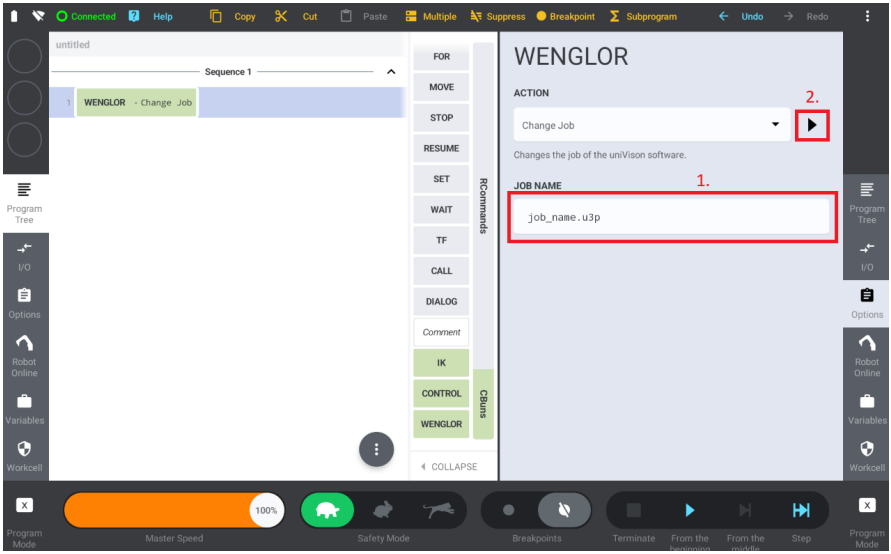
5.10.6 Robot Program

The wenglor device node

With the wenglor device you have access to all method calls of the robot vision API (see section “5.13 Generic Robot Vision API”).



All methods of the wenglor node provide a test option that executes the method without the need of a robot program. To execute the method, in this case a job change, enter the desired job name and hit the play button.



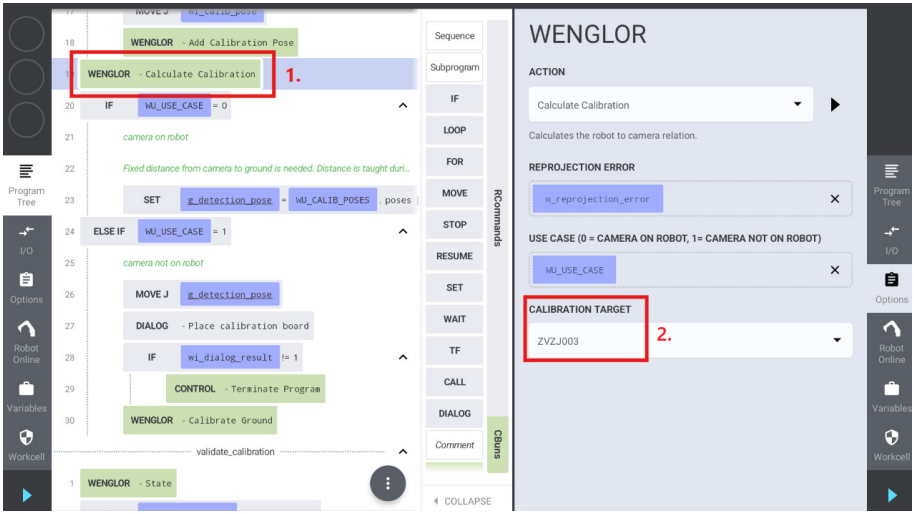
The robot program requires small adjustments depending on the use case.

Update calibration target

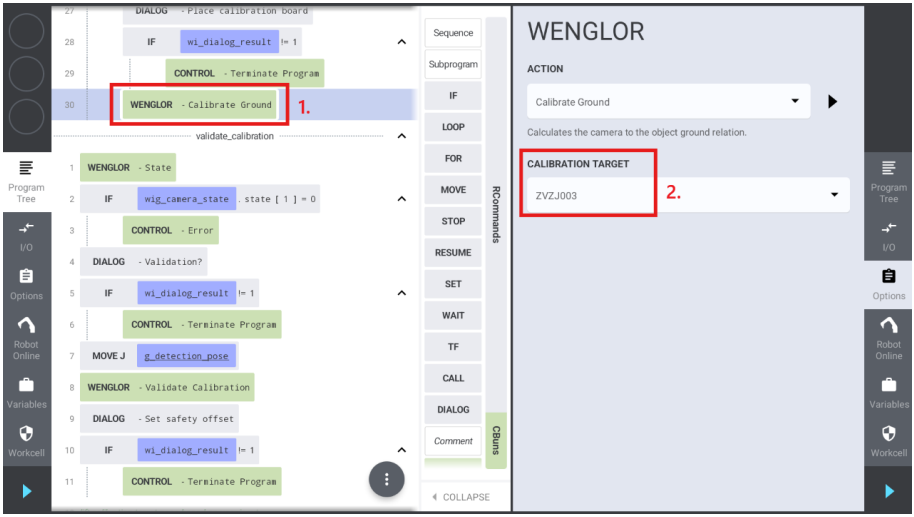
The following commands require the input of the calibration plate:

- Calculate Calibration
- Calibrate Ground
- Calibrate Target
- Get Target Pose

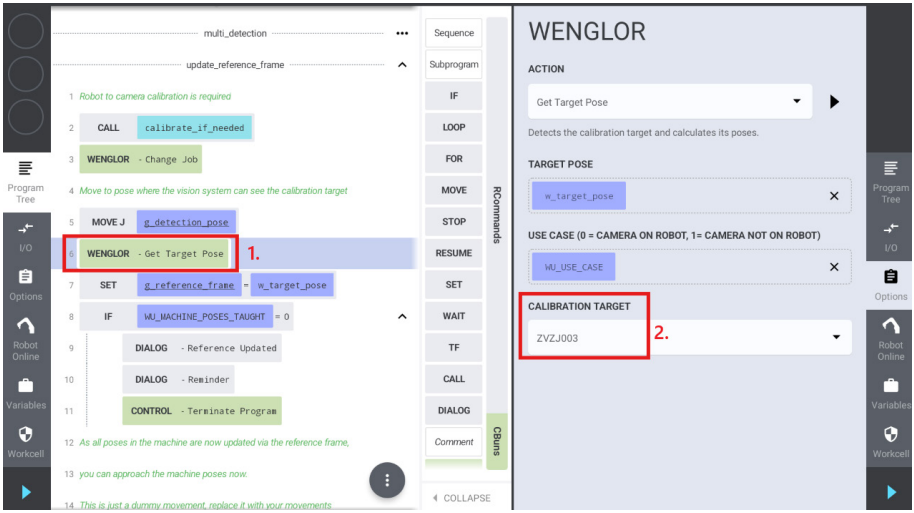
In the robot example program, the calibration target must be set in the run_calibration subprogram .



Set the calibration target at the second calibration step if the camera is not mounted on the robot (in run_calibration).



Set the calibration target in the update_reference_frame subroutine.

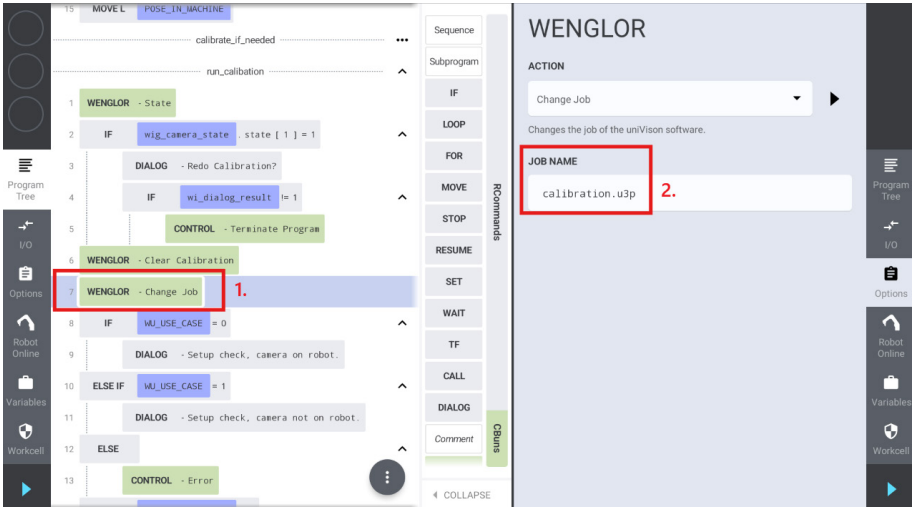


Update uniVision job names

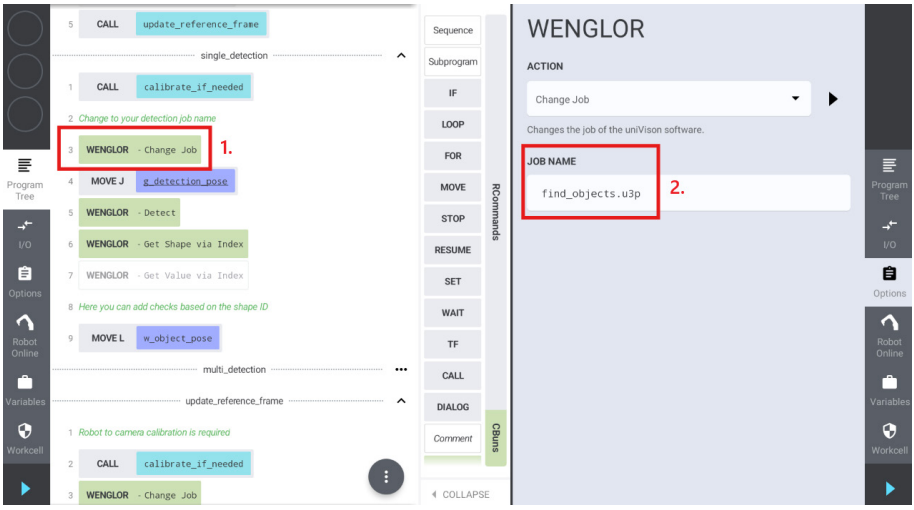
By default, the following uniVision job names are used:

- Calibration job: calibration.u3p
- Detection of objects: find_objects.u3p
- Detection of calibration target: find_target.u3p

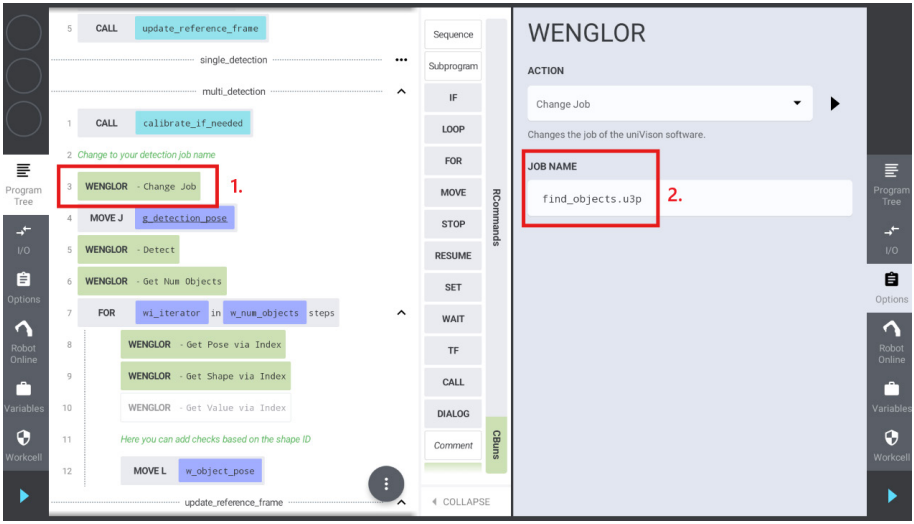
If using different uniVision job names, make sure to update run_calibration.



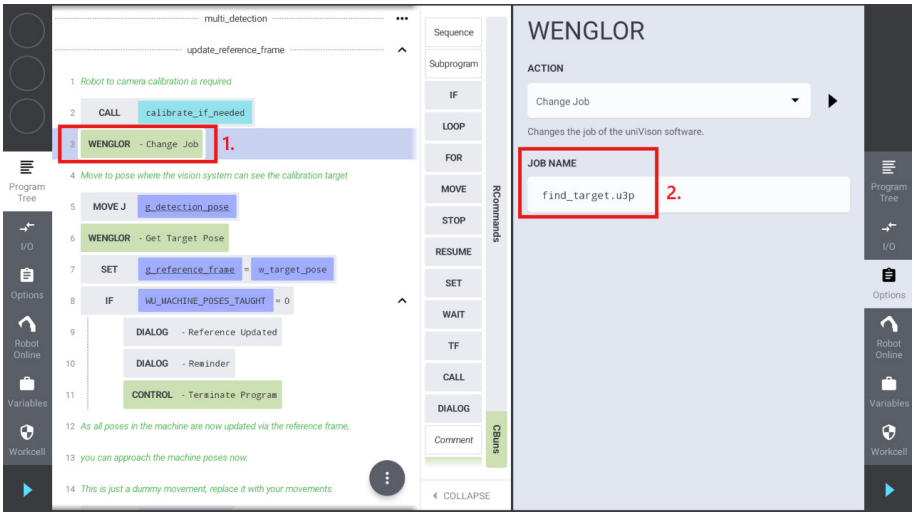
Adjust the uniVision job name at the single_detection subroutine.



Adjust the uniVision job name at the multi_detection subroutine.



Update the uniVision job name at the update_reference_frame subroutine.

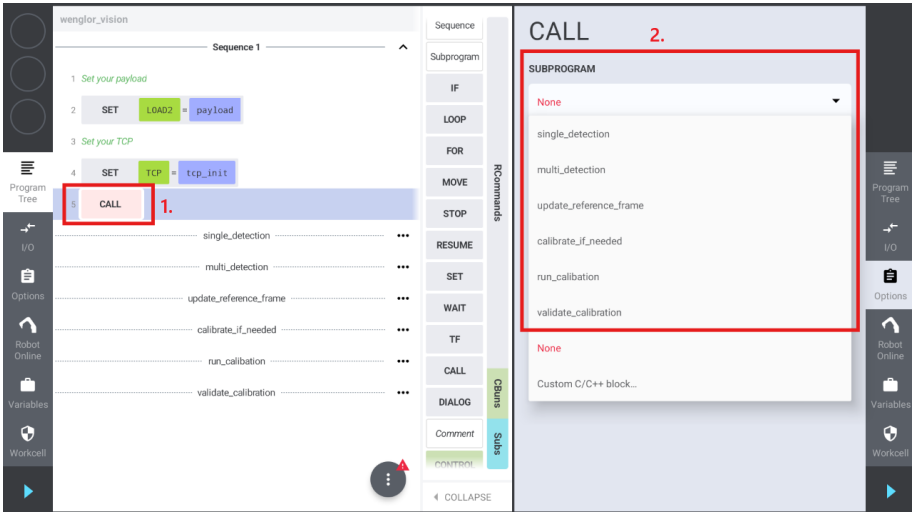


Run the program

After configuration, run the program. Several dialogs guide you through the process. The example program is just an example. Adjust it to your needs.

Subprogram name	Details
calibrate_if_needed	Checks the camera state for errors and the calibration state. In case of no available calibration, it calls the run_calibration subprogram. After the calibration program, it checks the state again to see if the calibration was successful.
single_detection	Calls calibrate_if_needed first. Loads the detection job, moves the robot to the detection pose and triggers the object detection. After this, the robot moves in a linear movement to the object. Even if multiple objects are detected, the subprogram will only move to one object.
multi_detection	Calls calibrate_if_needed first. Loads the detection job, moves the robot to the detection pose and triggers the object detection. Then it iterates through the object information via an index accessor and moves the robot to all of the objects.
update_reference_frame	Calls calibrate_if_needed first. Loads the find_target job and moves the robot to the pose where the camera can see the calibration target. Then it detects the calibration target pose and writes it into the persistent g_reference_frame variable. If the poses in the machine related to the reference frame are not taught yet, the program will stop. If you execute the subroutine again after setting WU_MACHINE_POSES_TAUGHT it will also move to the updated poses in the machine. You need to add your poses at the bottom of this subroutine. For details, see section “5.1 Basics with Robot Server”.
run_calibration	Loads the calibration job, clears the temporary calibration data (not the calibration itself) and moves the robot to the calibration poses. Then it calculates the calibration. If the camera is not mounted on the robot, it performs the second calibration step to calibrate the camera to object ground level.
validate_calibration	Checks the calibration results by moving the robot TCP to the bottom left corner of the calibration target. As the camera is not triggered again, it is important that the calibration target was not moved between the calibration and the validation. For safety reasons, enter a safety offset (in mm) before the actual movement. This safety offset shifts the target pose above the calibration target.

Select the subprogram to execute.

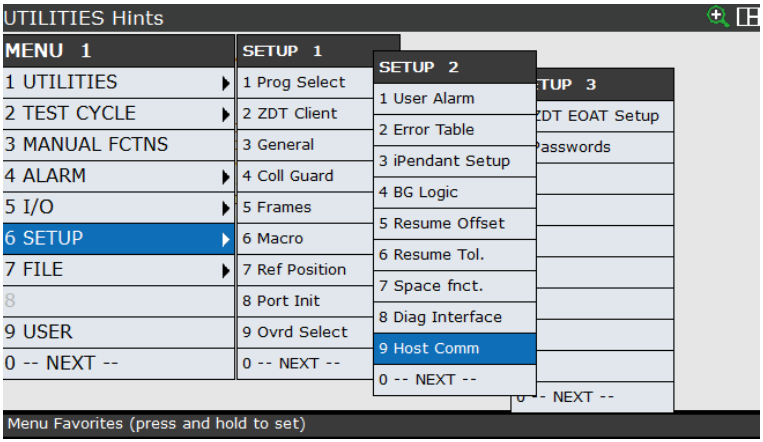


5.11 Fanuc Robots

The robot example program was tested with the FANUC R-30iB Mate Plus robot controller with the software V9.40 (Note: Make sure to use the same software version on the FANUC robot controller) and the LR Mate 200iD robot arm. It is available on https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

Network setup of robot controller

Adjust the network settings of the robot controller. Go to Menu → (6) Setup → Setup 2 → (9) Host Comm.

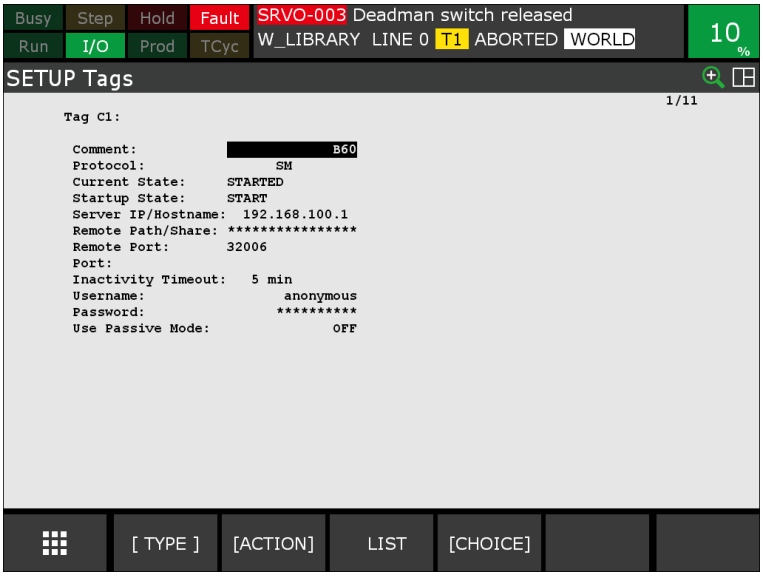


Select “TCP/IP” and enter the IP address of the robot controller (e.g. 192.168.100.11).

```
TCP/IP
    Robot name:                ROBOT
    Port#1 IP addr:    192.168.100.11
    Subnet Mask:        255.255.255.0
    Board address:
    Router IP addr:      192.168.100.1
    PC Jog IP addr:      192.168.0.100
    OP Panel IP addr:    192.168.0.100
```

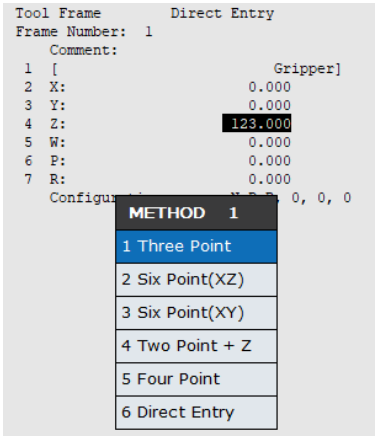
Socket messaging

To set the socket messaging client information, go to Menu → (6) Setup → (9) Host Comm. In this window, select Show (bottom bar) → Clients. Now select the Client you want to use. In the example, we use C1. Set the IP address of the Machine Vision Device (by default 192.168.100.1) and the port (by default 32006). For details, see socket messaging in the operating instructions of FANUC.



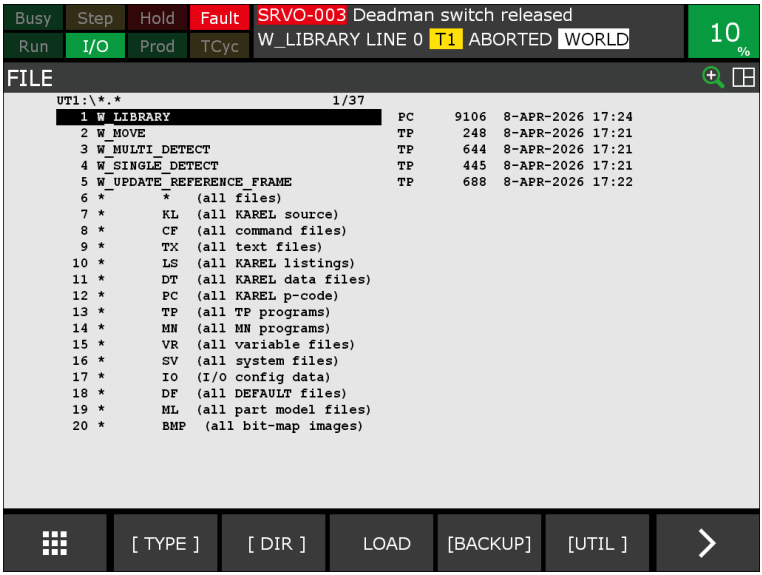
Tool setup

Go to Menu → Setup → Frames → Other (bottom bar) → Tool Frame. In this window, select the tool ID. Select Method (bottom bar) and pick the method for setting the TCP (e.g. Three Point).

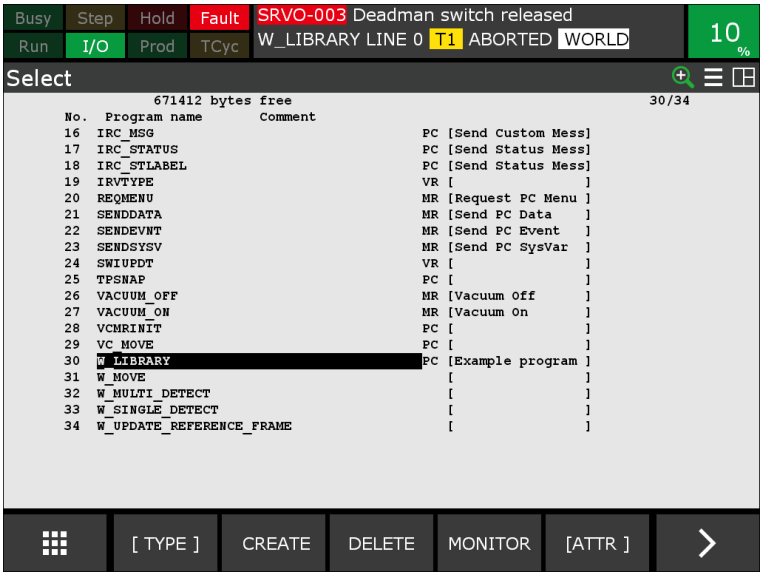


System variables
Check the system variables. Go to Menu → Next → System → Variables.
$KAREL_ENB needs to be set to 1 to enable working with Karel files. W_LIBRARY is a Karel file, so it is necessary.

Transfer files to robot
To transfer the files to the robot, use FTP (setup FTP similar to socket messaging) or a USB stick. For transferring the files via a USB stick, go to Menu → File → File → UTIL (bottom bar) → USB on TP (or Teach Panel Slot)/USB Disk (or Controller Slot) → Select and enter * (all files) → Copy (bottom bar).

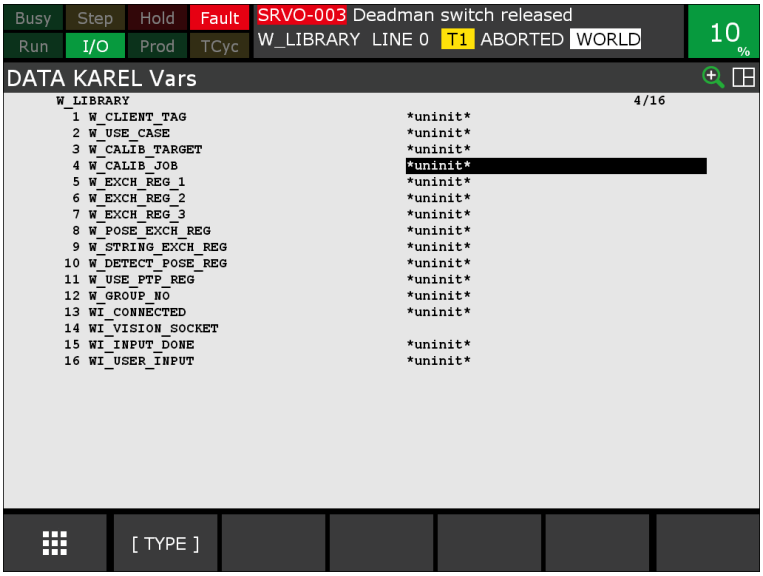


Select the target device Mem Device (MD) → Select DO_COPY (bottom bar).

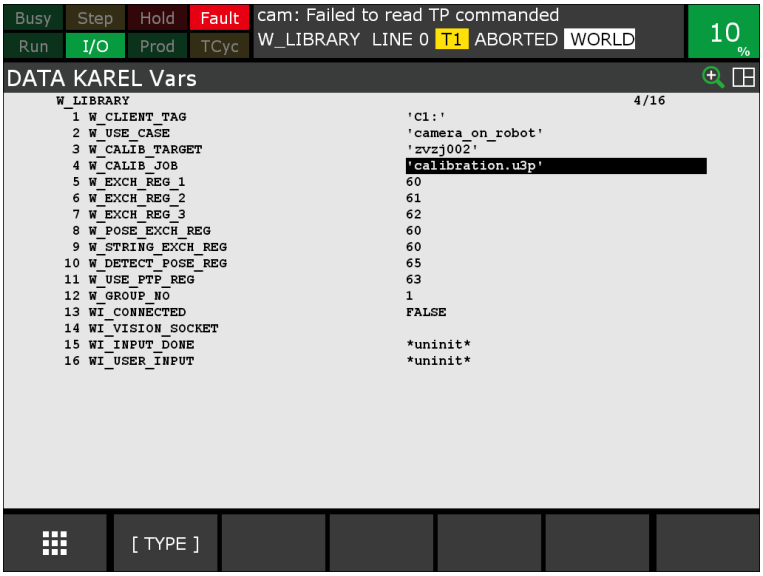


Karel program

The Karel program uses several variables. To access the Karel variables, select the corresponding PC file W_LIBRARY. Then go to Data → Type (bottom bar) → Karel Vars. At first, they are uninitialized.



To set the default values, run the Karel program W_LIBRARY. It will return a cam error, but it is fine in this case. Now you can adjust the variables to your use case. Consider that the Karel file needs to exchange registers to write the results of the commands. Please check if they conflict with your register usage and change them if required. Variables with a WI prefix are internal variables that are not meant to be changed.



User configurable Karel variables

The following variables (prefix w_) can be adjusted under Data → Karel Vars after selecting the Karel program W_LIBRARY. Variables with a wi_ prefix are internal and must not be changed.

Variable	Default	Note
w_client_tag	C1:	Host Comm client tag configured for the vision device (see Socket Messaging)
w_use_case	camera_on_robot	Either camera_on_robot or camera_not_on_robot
w_calib_target	zvzj002	ID of calibration target (zvzj001, zvzj002, zvzj003, zvzj004)
w_calib_job	calibration.u3p	uniVision job for calibration
w_exch_reg_1	60	Primary output register (R) for BOOLEAN / INTEGER / REAL
w_exch_reg_2	61	Secondary output register (R)
w_exch_reg_3	62	Tertiary output register (R)
w_pose_exch_reg	60	Output pose register (PR)
w_string_exch_reg	60	Output string register (SR)
w_detect_pose_reg	65	Detection pose register (PR) – keep in sync with the PR referenced by the TP programs

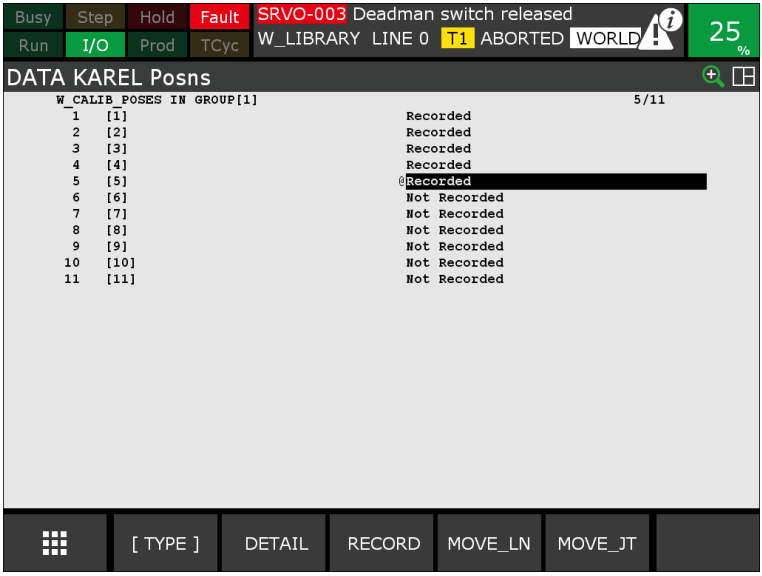
Variable	Default	Note
w_use_ptp_reg	63	Register (R) used to switch W_MOVE between PTP (1) and LIN (0)
w_group_no	1	Robot group number
w_calib_poses	-	Array of up to eleven XYZWPR calibration poses (minimum five required), set under → Data → Karel Pos)

Set the calibration and detection poses

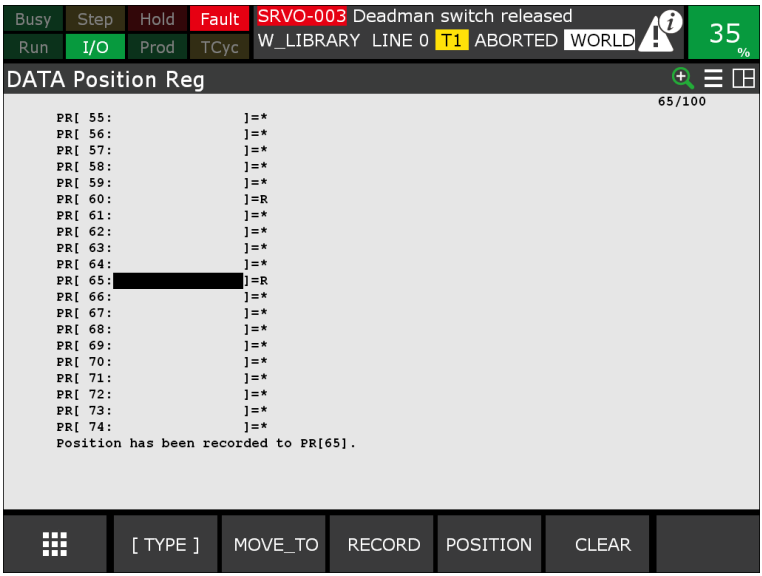
Set up maximum eleven calibration poses (at least five calibration poses). Depending on the use case, the detection pose is handled differently:

- Camera on robot: The detection pose is set during the calibration and also used for the validation.
- Camera not on robot: The detection pose needs to be set by the user. It is also used as a retreat pose after the calibration movement, in which you remove the calibration plate from the robot and place it on the object ground. It is also used for the validation.
- Both cases: Set the detection pose in a global register. By default, pose register 65 is used, as you can see in the provided example programs W_SINGLE_DETECT and W_MULTI_DETECT.

Select the Karel file W_LIBRARY → Data → Karel Pos → W_CALIB_POSES.

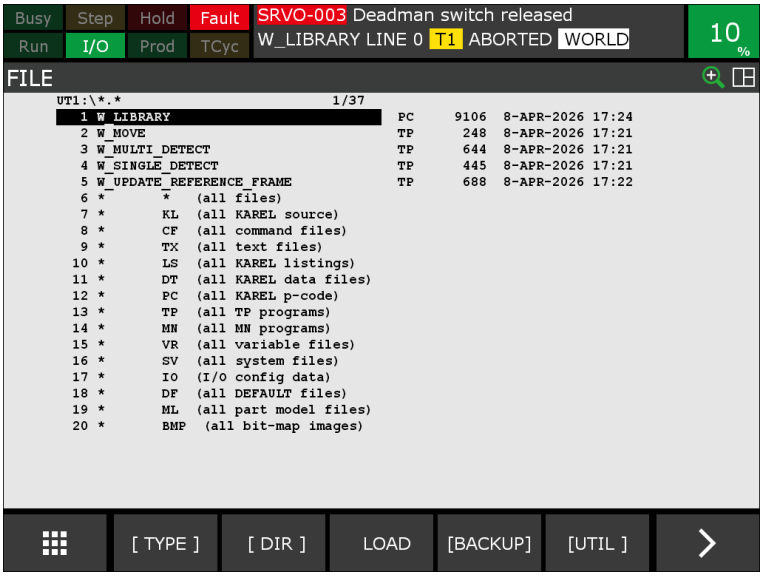


Set the detection pose. By default, pose register 65 is used. If PR65 is used for other use cases, you can also pick another PR, but you need to update the TP programs then. If you want to change the detection pose register, please also update the Karel variable.

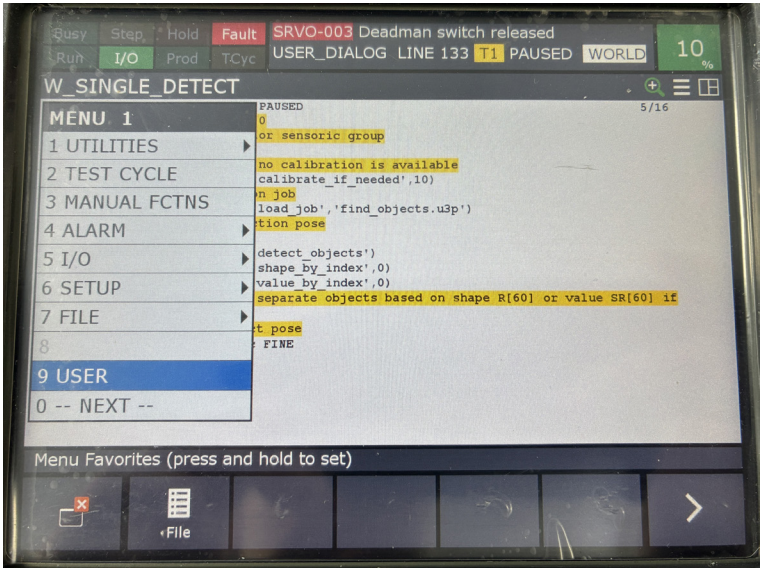


Calibration and validation process

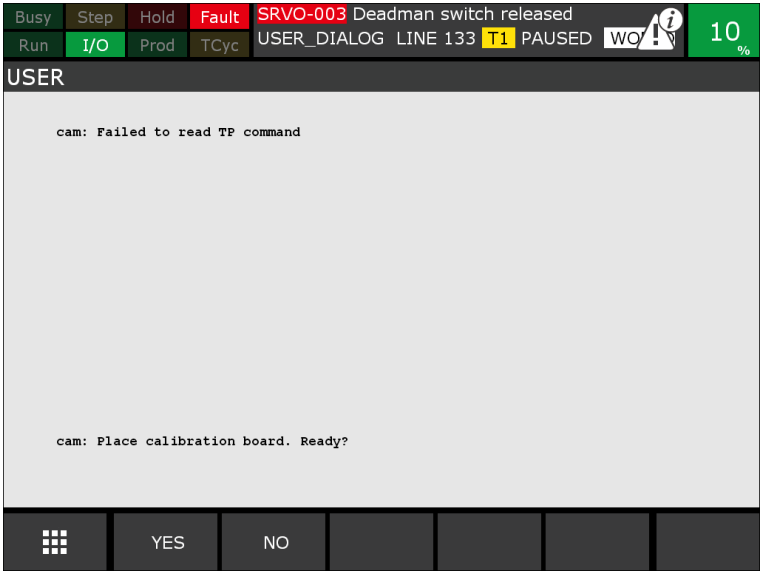
If there is no calibration file available on the Machine Vision Device, the calibration process is started by the example program. Select either W_SINGLE_DETECT, W_MULTI_DETECT or W_UPDATE_REFERENCE_FRAME.



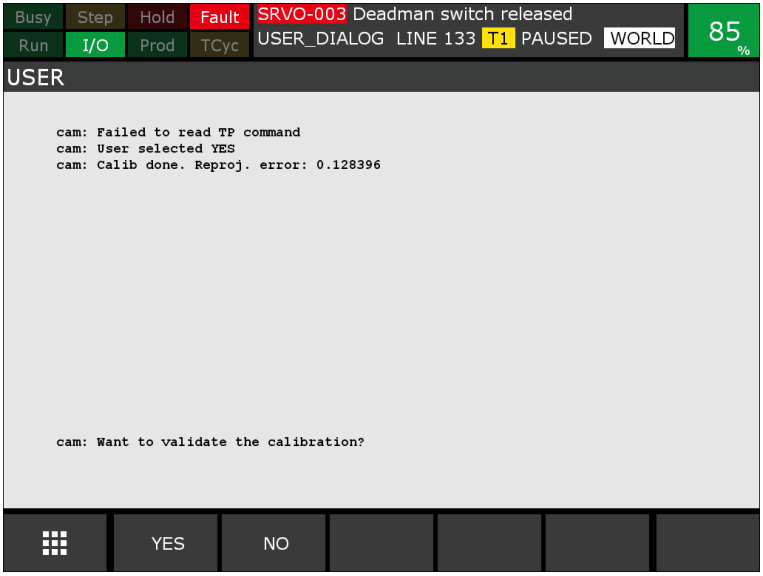
The calibration process uses user prompts. To see them, go to Menu → User.



Use SHIFT+F1 to put “YES”, SHIFT+F2 to put “FALSE”. The program needs to be active, so the program can read the user input. So in total the Deadman switch + SHIFT + F1 or F2 need to be pressed at the same time.



The program returns the registered user input. After this, the calibration movement starts. After the calibration is done, the reprojection error is returned.



For the optional validation, the robot will first move to the detection pose as a safe retreat pose and then move to the bottom left corner of the calibration plate. By default, a safety offset is applied (adjustable in the Karel variables). So the robot will move above the calibration plate.

Using the Karel routines from TP program

The Karel backend uses TP call parameters as inputs and returns its values to global registers. These are the default values (editable via the Karel variables when selecting the Karel program).

Register	Note
R[60] w_exch_reg_1	• Used for single and multiple output routines • Can contain BOOLEAN, INTEGER and REAL
R[61] w_exch_reg_2	• Only used if two or more outputs are required • Can contain BOOLEAN, INTEGER and REAL
R[62] w_exch_reg_3	• Placeholder for routines with more than two outputs • Can contain BOOLEAN, INTEGER and REAL
R[60] w_pose_exch_reg	Output pose like object pose or calibration target pose (validation)
SR[60] w_string_exch_reg	Output string like additional value or current uniVision job name

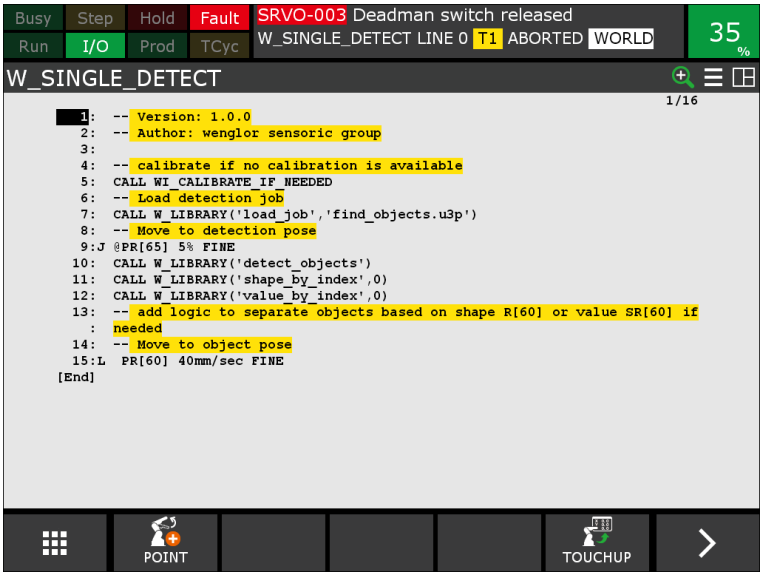
Callable Karel Routines with inputs and outputs

Routine Name	Input	Example	Output
cam_status		CALL W_LIBRARY('cam_status')	1. w_exch_reg_1 **1** (calibration data found) or **0** (no calibration data)
load_job	1. job name	CALL W_LIBRARY('load_job','find_objects.u3p')	
get_job		CALL W_LIBRARY('get_job')	1. w_string_exch_reg Name of active uniVision job
clear_calib_buffer		CALL W_LIBRARY('clear_calib_buffer')	
add_calib_pose		CALL W_LIBRARY('add_calib_pose')	
calc_calibration		CALL W_LIBRARY('calc_calibration')	1. w_exch_reg_1 The accuracy of the included intrinsic camera calibration
calib_to_ground		CALL W_LIBRARY('calib_to_ground')	
calib_to_target		CALL W_LIBRARY('calib_to_target')	
run_calibration		CALL W_LIBRARY('run_calibration')	

Routine Name	Input	Example	Output
validate_calibration	1. Safety offset in mm (REAL)	CALL W_LIBRARY('validate_calibration', 10.5)	
calibrate_if_needed	1. Safety offset in mm (REAL)	CALL W_LIBRARY('calibrate_if_needed', 10.5)	
detect_objects		CALL W_LIBRARY('detect_objects')	1. w_pose_exch_reg Object pose that is linked to uniVision Device Robot Vision Result List Index 0
detect_target		CALL W_LIBRARY('detect_target')	1. w_pose_exch_reg Calibration target pose
num_objects		CALL W_LIBRARY('num_objects')	1. w_exch_reg_1 Number of found objects
pose_by_index	1. Index of object	CALL W_LIBRARY('pose_by_index',R[11])	1. w_pose_exch_reg Object pose for given index
value_by_index	1. Index of object	CALL W_LIBRARY('shape_by_index',R[11])	1. w_string_exch_reg Additional value for object with given index
shape_by_index	1. Index of object	CALL W_LIBRARY('value_by_index',R[11])	1. w_exch_reg_1 Shape model ID for object with given index

Set up the TP program

Set up your TP program with the described routines. You can also have a look at the provided example programs W_SINGLE_DETECT, W_MULTI_DETECT or W_UPDATE_REFERENCE_FRAME. For W_SINGLE_DETECT it looks as follow:



5.12 Generic Robot Vision Interfaces

Select the robot manufacturer “Generic” on the device website (tab “Jobs” → "Robot Server”, see section “5.2 Settings on Device Website”) in order to use the generic string based robot vision API (see section “5.13 Generic Robot Vision API”).



NOTE!
The python robot example for the generic string based robot vision API is available on www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

5.13 Generic Robot Vision API

The generic robot vision API describes the communication between the robot controller and the wenglor robot server on the Machine Vision Device.

Overview of commands for string and XML based robots:

Command	Description	Unit/Note
job:change[job_name.u3p];	Changes the uniVision job of the processing instance to the given job.	File ending “.u3p” is mandatory. Don't wrap the job name in quotes
job:get;	Requests the currently active job of the processing instance.	String
calibration:clear;	Clears the internal buffer of the robot server. Required in case of re-calibrating.	
calibration:add[pose_information];	Triggers and saves the current camera image together with the given robot pose in the internal robot server buffer.	Pose information in [x, y, z, rx, ry, rz] x, y, z in meter rx, ry, rz as rotation vector in radiant
calibration:calculate[calibration_case, calibration_target];	Calculates the hand-eye calibration based on the saved calibration poses and images from the internal robot server. The results are stored on the Machine Vision Device.	Calibration case options: camera_on_robot camera_not_on_robot Calibration target options: ZVZJ001 ZVZJ002 ZVZJ003 ZVZJ004 Comment: For ZVZJ005 use ZVZJ001 and for ZVZJ006 use ZVZJ002 (same size).

Command	Description	Unit/Note
calibration:ground[calibration_target];	Triggers the uniVision job, calculates and saves the current calibration target pose as an object ground reference.	Calibration target options: ZVZJ001 ZVZJ002 ZVZJ003 ZVZJ004 Comment: For ZVZJ005 use ZVZJ001 and for ZVZJ006 use ZVZJ002 (same size).
state[calibration_case];	Requests the state of the robot server. The first bit represents the overall state. The second bit represents if a calibration is present for the provided use case or not.	Calibration case options: camera_on_robot camera_not_on_robot First bit: Connection error state 0: No camera error present 1: Camera is in error state Second bit: Calibration state 0: No calibration present 1: Calibration data could be read Example: 01 0: No camera error 1: Calibration available Example: 10 1: Camera in error state 0: Calibration not available Example: 11 1: Camera in error state 1: Calibration available
detect[calibration_case, pose_information];	Triggers the uniVision job, sends the device robot vision data to the robot server. There the 3D object pose is calculated based on the current calibration data and returned to the robot.	Calibration case options: camera_on_robot camera_not_on_robot Pose information in [x, y, z, rx, ry, rz] x, y, z in meter rx, ry, rz as rotation vector in radiant Object pose (array of floating point numbers)
num_objects:get;	Requests the number of objects available in the robot server buffer	Positive number

Command	Description	Unit/Note
validate[calibration_case, detection_pose];	Provides the calibration target origin 3D pose (bottom left target) based on the information saved during the calibration.	Calibration case options: camera_on_robot camera_not_on_robot Pose information in [x, y, z, rx, ry, rz] x, y, z in meter rx, ry, rz as rotation vector in radiant Calibration target pose (array of floating point numbers)
pose:get[index];	Requests the object pose with the given index from the robot server buffer. Detect must be called first to fill the buffer.	Index starts at 0 Pose information in [x, y, z, rx, ry, rz] x, y, z in meter rx, ry, rz as rotation vector in radiant Object pose (array of floating point numbers
shape:get[index];	Requests the shape model with the given index from the robot server buffer. Detect must be called first to fill the buffer.	Index starts at 0 Number
value:get[index];	Requests the additional value with the given index from the robot server buffer. Detect must be called first to fill the buffer.	Index starts at 0 String

Command	Description	Unit/Note
target:pose[calibration_case, calibration_target, pose_information];	Triggers the uniVision job, sends the device robot vision data to the robot vision server. There the 3d pose for the calibration target is calculated based on the current calibration data and returned to the robot. It also fills the additional value internally which can be accessed via value:get[0]	Calibration case options: • "camera_on_robot" • "camera_not_on_robot" Calibration target options: • ZVZJ001 • ZVZJ002 • ZVZJ003 • ZVZJ004 Comment: For ZVZJ005 use ZVZJ001 and for ZVZJ006 use ZVZJ002 (same size). Pose information in [x ,y ,z ,rx, ry, rz] x, y, z in meter rx, ry, rz in Rodrigues convetion in radiant target_pose (array of floating point numbers)
calibration:target[calibration_case, calibration_target]	Calibrates the camera to target relation without creating a new calibration file. The new calibration data is just cached internally in the camera.	Calibration case options: • "camera_on_robot" • "camera_not_on_robot" Calibration target options: • ZVZJ001 • ZVZJ002 • ZVZJ003 • ZVZJ004 Comment: For ZVZJ005 use ZVZJ001 and for ZVZJ006 use ZVZJ002 (same size).

Commands and return values for string robots:

Command	Successful return value (for string based robots)	Error return value (for string based robots)
job:change[job_name.u3p];	0	Error code as negative number
job:get;	uniVision job name [string]	Error code as negative number
calibration:clear;	0	Error code as negative number

Command	Successful return value (for string based robots)	Error return value (for string based robots)
calibration:add[pose_information];	0	Error code as negative number
calibration:calculate[calibration_case, calibration_target];	reprojection error positive floating point	Error code as negative number
calibration:ground[calibration_target];	0	Error code as negative number
state[calibration_case];	Two binary numbers • First bit: Camera error • Second bit: Calibration state	
detect[calibration_case, pose_information];	Object pose (array of floating point numbers)	Error code as negative number
num_objects:get;	Number of objects available in robot server (positive number)	Error code as negative number
validate[calibration_case, detection_pose];	Calibration target pose (array of floating point numbers)	Error code as negative number
pose:get[index];	Object pose to the object with the corresponding index (starting with 0) (Array of floating point numbers)	Error code as negative number
shape:get[index];	Shape model linked in Device Robot Vision to the object with the corresponding index (starting with 0) (number)	Error code as negative number
value:get[index];	Additional value linked in Device Robot Vision to the object with the corresponding index (starting with 0) (string)	Error code as negative number
target:pose[calibration_case, calibration_target, pose_information];	target_pose (array of floating point numbers)	Error code as negative number
calibration:target[calibration_case, calibration_target]	0	Error code as negative number

The following list contains the return value error codes:

Code number	Error message	Notes
-5001	“General Error”	If an error was recognized, but could not be assigned to any other error code.
-5002	"Badly formatted request"	• If the extraction of the parameter failed. • The command ending is missing. • The command is unknown.
-5003	"No connnection to univision"	• The required connection to the processing instance is not available. • The required connection to Device Robot Vision is not available.
-5004	"Unknown univision job name"	If the job could not be loaded, but for a different reason than a connection issue.
-5005	"Badly configured univision job"	If the number of shape models of locator/pattern match does not fit to the configuration of Device Robot Vision.
-5006	"Calibration failed"	If the calibration failed for any reason.
-5007	"No calibration data"	If the required calibration data is not available to calculate the 3d object pose.
-5008	"No object found"	If Device Robot Vision sends result true count = 0.
-5009	"Bad or empty device robot vision message"	If the extraction of Device Robot Vision message fails, mostly the case if it is empty.
-5010	“Index error”	If the user tries to access data that is out of range, e.g. num_objects:get; returned 1 object was found, and you are trying to access shape:get[1];. As the index starts counting at 0, the user would try to access object 2, while only one was found.

The answer from the Machine Vision Device differs between string and XML based robots. At XML based robots, the robot server returns always the complete XML frame to the robot where only the request related tag is filled with information, while the other tags are set to default values. For example, the XML answer to the command job:get; looks accordingly.

```
<sensor>
    <reply>
        <command>job:get;</command>
        <return_code>0</return_code>
        <job_name>calibration.u3p</job_name>
        <calib_accuracy>0</calib_accuracy>
        <state>00</state>
        <object_pose X="0" Y="0" Z="0" RX="0" RY="0" RZ="0"/>
        <num_objects>0</num_objects>
        <shape_model>0</shape_model>
        <additional_value></additional_value>
    </reply>
</sensor>
```

The following table shows examples of the reply for string based robots (e.g. ABB).

Command from robot	Successful reply from robot server
job:change[calibration.u3p];	0
job:get;	calibration.u3p
calibration:clear;	0
calibration:add[[0.021476,-1.344395,0.716645, -3.080280,-0.024544,-0.476923]];	0
calibration:calculate[camera_not_on_robot,zvzj004];	0.129936
calibration:ground[zvzj004];	0
state[camera_not_on_robot];	01
detect[camera_not_on_robot,[0.047871, -0.856617,0.830479,0.874365,3.004280,-0.045222]];	(0.048705,-0.094640, -0.110581,0.102101, -3.131079,0.003192)
validate[camera_not_on_robot, [0.047871,- 0.856617,0.830479,0.874365,3.004280,-0.045222]];	(0.318205,-0.164630, -0.100281,0.090210, -1.131079,1.003192)
num_objects:get;	5
pose:get[0];	(0.048705,-0.094640, -0.110581,0.102101, -3.131079,0.003192)
shape:get[1];	0
value:get[4];	0.903153
target_pose[camera_on_robot,zvzj001,[0.047871,-0.856617, 0.830479,0.874365,3.004280,-0.045222]]	(0.318205,-0.164630,-0.100281,0.090210,-1.131079,1.003192)
calibration:target[camera_on_robot,zvzj001]	0

NOTE!
An example robot program structure written in Python that shows how to use the generic robot vision API is available on https://www.wenglor.com/product/DNNF023 → Downloads → Programming examples and configuration files → Examples_Robot_Vision.

6. Change Index of Operating Instructions

Version	Date	Description	Associated software version
1.0.0	10.03.2025	Initial version	Software uniVision 3.4.0
1.1.0	13.05.2025	Updated description for the robot examples	Software uniVision 3.5.0
1.2.0	01.09.2025	Updated description for the new features of uniVision 3.6.0	Software uniVision 3.6.0
1.3.0	13.04.2026	Updated description for the new features of uniVision 3.7.0	Software uniVision 3.7.0