3D Robot Calibration
Command Manual

This document aims to explain how to properly use the 3D Robot Calibration command to implement simple and complex robot engineering tasks with EyeVision.

0 Table of Contents

Last changed: 23.09.2021 (nwe)

1 Introduction

The key problem of robot operation with 3D computer vision is the correct communication of positioning information. Since many internal rotations and translations between the robots base, its hand, its tool center and the camera origin are present, this often no trivial task. A lot of these offsets, which are 6DoF rotation translations, are usually unknown and hard or impossible to calculate by hand.

The 3D Robot Calibration command in EyeVision provides a tool set for exactly this set of problems. It simplifies the operation of a robot together with a 3D camera as much as possible while still remaining universal for all use cases.

The essence of this document is to demonstrate the functionality of the 3D Robot Calibration command. All other EyeVision commands and program structures mentioned are only one of many possible examples and do not describe the only possible way to use and integrate 3D Robot Calibration command into the project.

This document is also accompanied by two example EyeVision projects. One for fixed camera and one for moving camera setup.

These can be downloaded with the following links:

3DRobotVision__RobotCalibration_FixedCamera.zip

3DRobotVision__RobotCalibration_MovingCamera.zip

2 Terminology and Basics

(6DoF) **Pose**

This describes a position and rotation of an object in the Cartesian space with 6 degrees of freedom (DoF), 3 position and 3 rotation float values. Not more and not less than exactly these 6 values are needed to fully describe the positioning of objects, robot hand positioning, coordinate centers, transformations, or offsets.

Rotation-Translation

This is a synonym for the (6DoF) Pose. In EyeVision the Poses are handled as the RotationTranslation Geo3D data type, which can be imported, exported, transformed and stored as such. It is for instance the output of 3D Object Match and input to Robot Communication command.

Robot Base

This is the center of the robots coordinate system. It is usually, but not necessarily near the mounting of the robot to the table surface. Its exact position and rotation is often unknown. In terms of coordinate systems, Robot Base describes the 6DoF Pose of the origin, i.e. not only its position but also its rotation.

It is in theory possible to change the Robot Base position inside the ROS. This can be done without harm, as long as this setting is done initially once and does not change after the calibration process. All robot hand poses must then be interpreted relative to this custom origin consistently at all times.

Robot Hand

This is the part of the robots mechanics, whose 6DoF Pose is controlled by the user/ROS. After sending a 6DoF Pose to the robot, the Robot Hand will be in exactly this Position and Rotation relative to the Robot Base. Its exact position and rotation is often unknown. Another name is “end-effector”.

The Robot Hand is not directly visible, but hidden in the mechanics. There is also an offset between the Robot Hand and the location of target objects when the hand grabs them.

Euler Angle rotation convention

Rotations in EyeVision are handled with 3 Euler angles. The Euler angle convention is the extrinsic Z→Y→X rotation order. For more information please read the last chapter of this document.

Transformation

Yet another name for Rotation-Translation. It is mostly used in the context of coordinate system modification.

3 Prerequisites and setup

Before going into calibration of the robot, some basic requirements on hardware and software side are to be met

3.0 EyeVision

For Robot Operation, EyeVision version V4.0.13 or above is necessary. All 3D Commands mentioned in the following should be licensed.

The user must be familiar with basic program editor functionality. This includes the use of sub-programs, pickup-lists and Geo3D data register.

The user must also be familiar with 3D Object pose detection using 3D commands like 3D Blob, 3D Object Match.

It is also recommend to know how to import/export Geo3D data from/into files,

3.1 Robot control

The robot hand pose must be controllable via any communicate protocol from inside EyeVision. This may either be TCP/IP, Serial or manual user interaction.

The communication is done in the form of sending 6DoF Poses in the form of Rotation-Translations for the robot hand relative to the Robot Base. All values are in mm.

The coordinate system of the robot must be Cartesian and Right-Handed.

If the robot uses a different rotation notation than extrinsic $Z \rightarrow Y \rightarrow X$ (EyeVision standard), a respective conversion procedure must be implemented for sending/receiving. For more information see the last chapter of this document.

It is recommended to create an EyeVision sub-program which reads Rotation-Translations from the Geo3D register, assembles a communication string, sends it to the robot and waits until all movement is done.

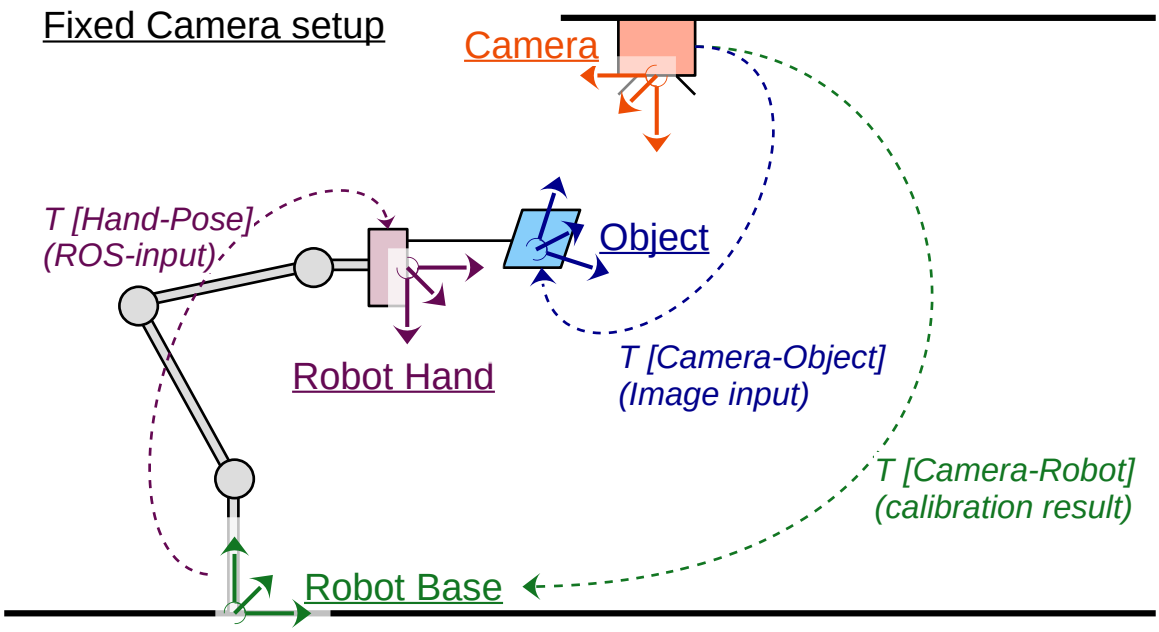
3.2 Camera

The 3D Imaging camera needs to be connected to the EyeVision software. The correct VIC must be selected in the hardware configuratior. Image capturing with the Capture Image command must work.

The camera must deliver calibrated 3D point clouds which are in mm (according to robot units) and Right-Handed.

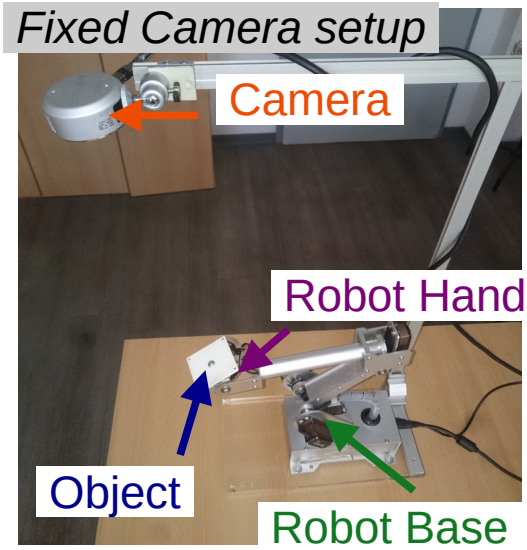
The precision of the camera must be better than the desired robot application precision.

4 Fixed Camera to Robot calibrations



This section covers the calibration process of the fixed camera case, which is sketched above. The Camera mount is fixed in space relative to the Robot Base and does not move during operation at any time.

The ultimate goal of the calibration is to find the Transformation from Camera to Robot Base. With the latter given, the point cloud origin pose can be set from Camera origin to the Robot Base.



For the calibration process a well visible object needs to be attached to the robot hand. Then the robot hand is moved to several positions. Each time the camera takes one image of the object. A simple evaluation program in EyeVision then extracts the object pose in Camera coordinates. Afterwards the calibration can be performed.

A step by step walk-through is now given in the following.

4.0 Calibration Object

A calibration object needs to be attached to the robots hand. It must be possible to detect the center of the object reliably with the usual 3D EyeVision tool set. If a unique rotation pose can be detected, this is of even greater advantage.

Possible objects could be balls, squares (for position only) or triangles, polygons, punctured shapes (for position + rotation).

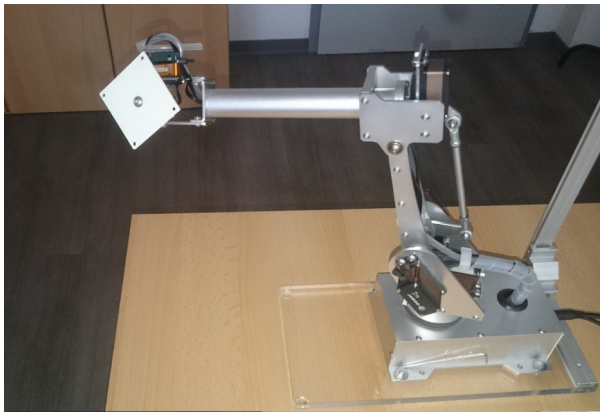
4.1 Robot Hand calibration poses

A set of several, different robot hand poses needs to be prepared. These can be any arbitrary poses as long as the robot hand can perform the movement and the camera can see the object in the robot hand in the respective pose.

If the calibration object can provide stable 6DoF pose estimation, including unique rotation, at least 3 poses are needed. If it cannot provide rotation information, at least 6 poses are needed.

The rotations of the poses should vary as much as possible. If they are all too similar, the transformation calculation might fail due to insufficient input information.

Two example poses A, B are shown in the images below.



It is recommended to save all these poses as a text file in a format, that EyeVision can import it as Rotation-Translation. The JSON format looks like the following:

```
{ "type": "3d.rotationtranslation",
  "position": { "x": -100.0, "y": 150.0, "z": 230.0 },
  "angle": { "a": 0.0, "b": -35.0, "g": 90.0 } }
```

where *a*, *b*, *g* denote rotation rotation angles in degrees around the X, Y, and Z axis respectively.

Another option is CSV format with the ordering type, x, y, z, a, b, g, like:

```
rotationtranslation, -100.0, 150.0, 230.0, 0.0, -35.0, 90.0
```

There should thus be one text file on the hard drive for each pose.

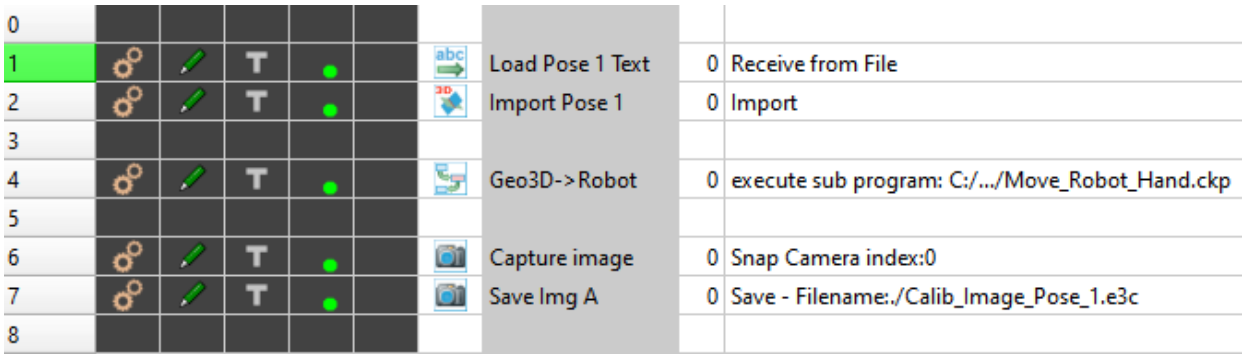
4.2 Capturing calibration images

Now an image of the camera needs to be taken for each pose. For this open EyeVision and assemble the program according to the following steps. It is recommend to use an individual program only for capturing the images.

For each pose the steps look like the following:

1. Load the text from the robot hand pose file into the global string using the Global String command.
2. Import the Rotation-Translation using the 3D Geometry command.
3. Add the Subprogram command and select the robot hand pose communication program. It should send the Rotation-Translation from Geo3D register position 0 to the robot.
4. Capture a point cloud with the Capture Image command.
5. Save the point cloud with the Capture Image command under an individual filename corresponding to the pose.

For one single pose this would look like this in the program editor:



After execution, one .e3c point cloud file per pose must be present on the hard drive.

4.3 Finding object pose from images

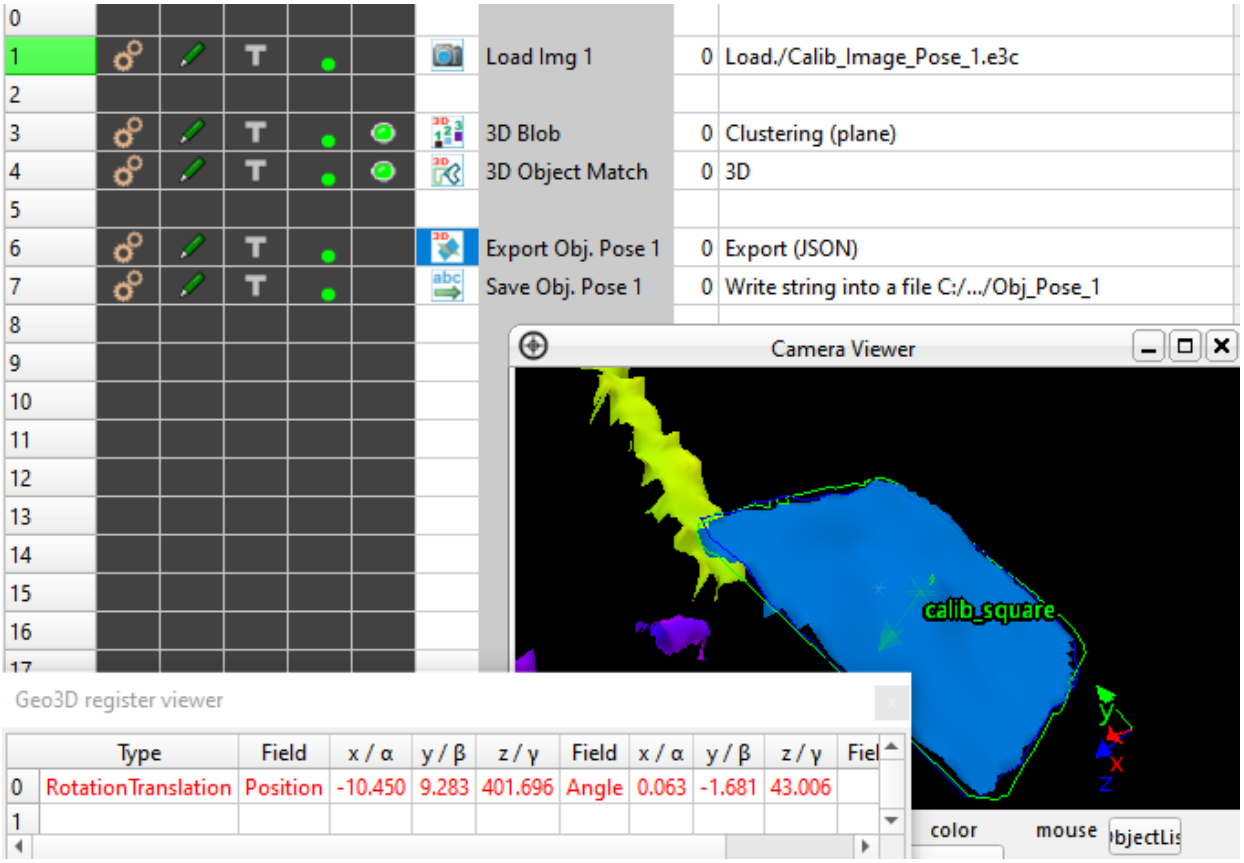
The next step is to perform the object detection and thus acquire the Camera-To-Object pose. The exact approach is highly dependent on the object and camera used. Thus the following is only an example. Again it is best to use an individual program for this task.

For each pose do the following step:

1. Load the image.
2. Perform an object plane clustering the 3D Blob command. Adjust the filter settings to help with object detection.

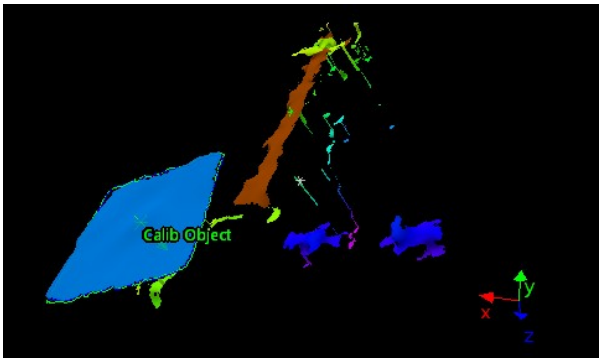
3. Match the calibration object shape to the objects with the 3D Object Match command to find the desired pose.
4. Export the resulting Rotation-Translation or Position of the object as text using the 3D Geometry command.
5. Write the text to a file with respective file name using the Global String command.

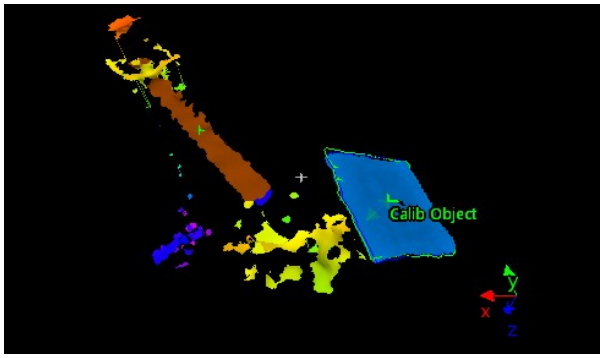
The program editor could look like this (for one single pose):



Note: The Object pose output here can be either Rotation-Translation or Point. If the object allows unique rotation estimation, than it is advised to save Rotation-Translation. If the object cannot provide rotation information (e.g. a ball, simple square) than position must be used as output type for saving. In the latter case it is advised to also use at least 9 poses. If needed, use the 3D Geometry command to extract the position from a Rotation-Translation object.

For the two poses from above this would look like the following:





4.4 Final Calibration

In this step the Robot Hand pose data and Camera-to-Object pose data from the last two sections are used together to calibration the camera to the robot.

Again it is advised to create an individual program for the following steps.

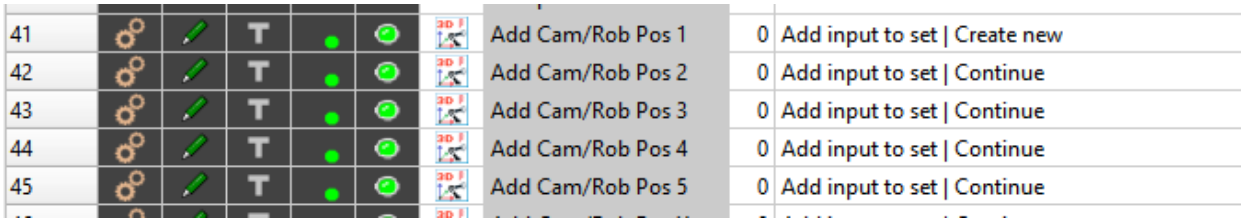
Assembling the program

For each pose:

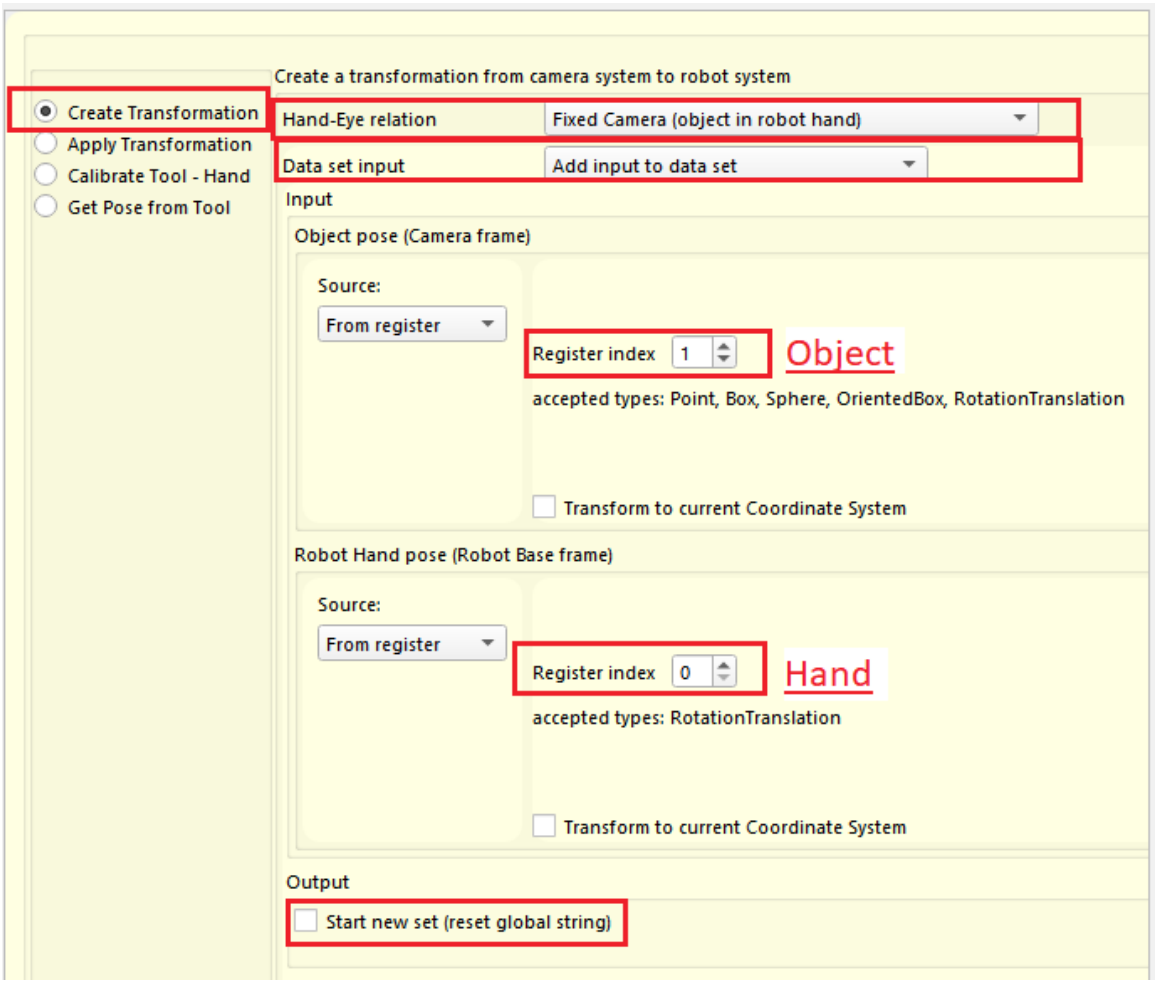
1. Load the object pose text from the hard drive with the Global String command.
2. Import the data into the 3D Register using the 3D Geometry command
3. Load the robot hand pose text from the hard drive with the Global String command.
4. Import the data into the 3D Register using the 3D Geometry command

0									
1							Load Obj. Pose 1	0	Receive from File
2							Import Obj. Pose 1	0	Import
3							Load Hand Pose 1	0	Receive from File
4							Import Hand Pose 1	0	Import
5									
6							Load Obj. Pose 2	0	Receive from File
7							Import Obj. Pose 2	0	Import
8							Load Hand Pose 2	0	Receive from File
9							Import Hand Pose 2	0	Import

Then place one 3D Robot Calibration command per pose below all imports commands.



Set the parameters according to the following image:

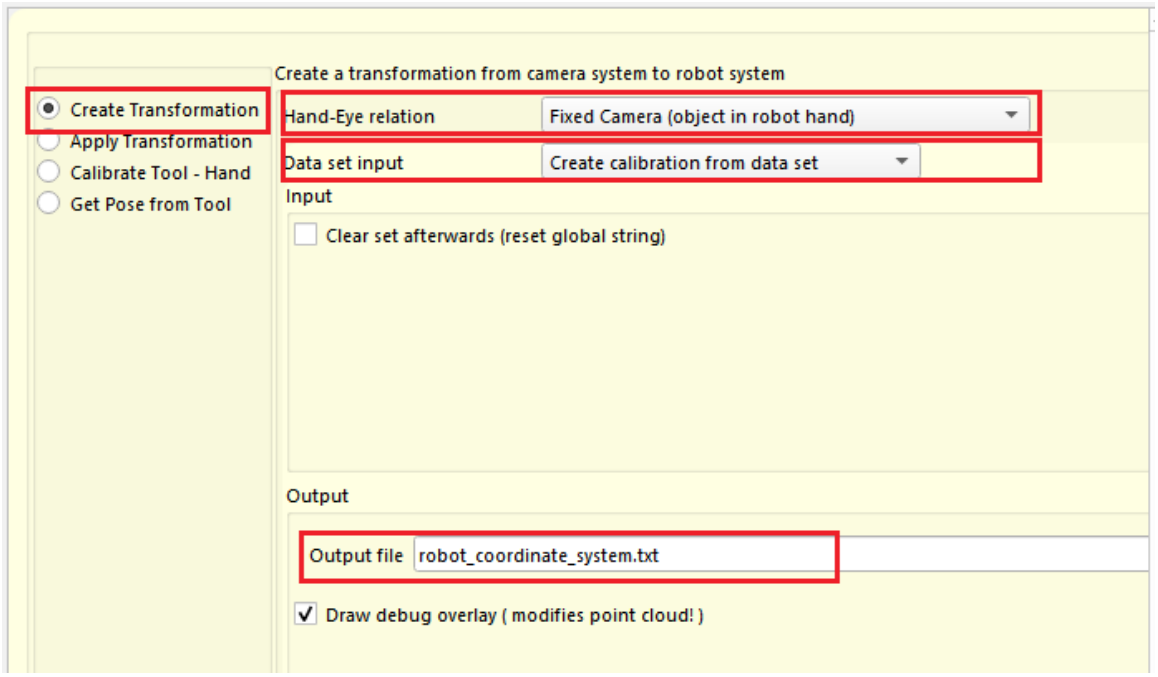


However, the Register index is different for each of these command instances since they refer to specific data positions in the Geo3D Register.

In this example the first command has Object index 1 and Hand index 0. The second has Object index 3 and Hand index 2. Then 5 and 4, and so on.

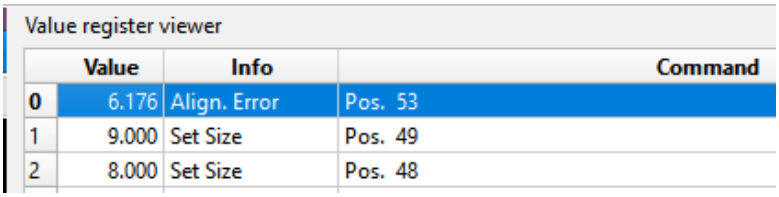
The first 3D Robot Calibration command, but only the first, should also have the *Start new set* option activated.

Now at the end of the program one final 3D Robot Calibration command instance is needed with the following settings:



Results and Output

After now running the whole program once, the value register should contain the mean alignment error of the calibration.

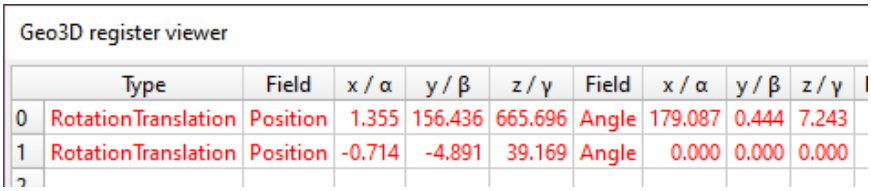


	Value	Info	Command
0	6.176	Align. Error	Pos. 53
1	9.000	Set Size	Pos. 49
2	8.000	Set Size	Pos. 48

This value describes the mean distance in mm between the measured object position by the camera and the predicted object position by the robot, given the calibration result. It can be seen as a measure of uncertainty for the calibration and should be as low as possible. If it is higher than the desired robot hand positioning precision, the calibration process must be revisited.

The Transformation (Rotation-Translation) from camera origin to robot base is written to the *Output file*. This is the *calibration file* which will be used later to apply the calibration.

The Geo3D Register (and pickup list) receive two Rotation-Translation objects as a result of the 3D Robot Calibration command:

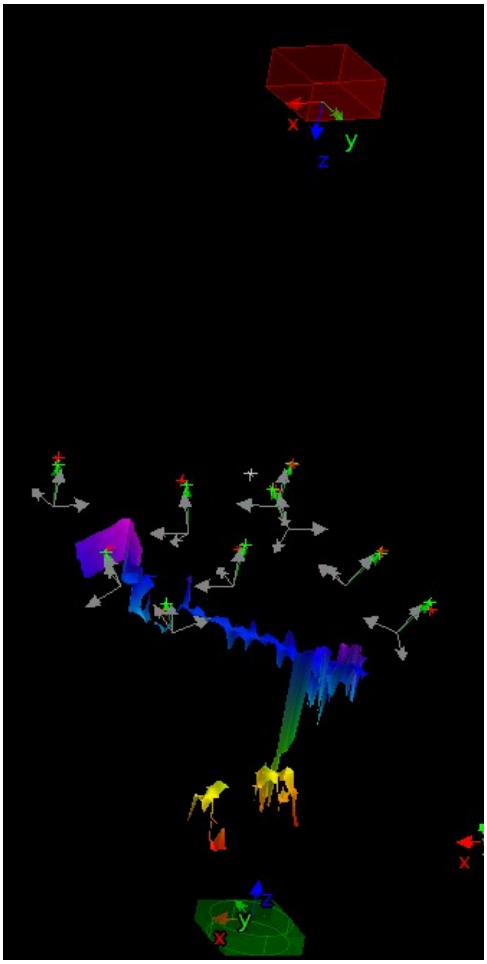


	Type	Field	x / α	y / β	z / γ	Field	x / α	y / β	z / γ
0	RotationTranslation	Position	1.355	156.436	665.696	Angle	179.087	0.444	7.243
1	RotationTranslation	Position	-0.714	-4.891	39.169	Angle	0.000	0.000	0.000

At index 0 is the camera origin to robot base transformation, which is also saved to the output file.

At index 1 is the robot hand to object offset, which is estimated as a by product. It is not saved anywhere and not necessary for the future. It can however serve as a cross-check to judge if the calibration is reasonable. If no object rotation was provided as input, than the rotation fields are set to 0.

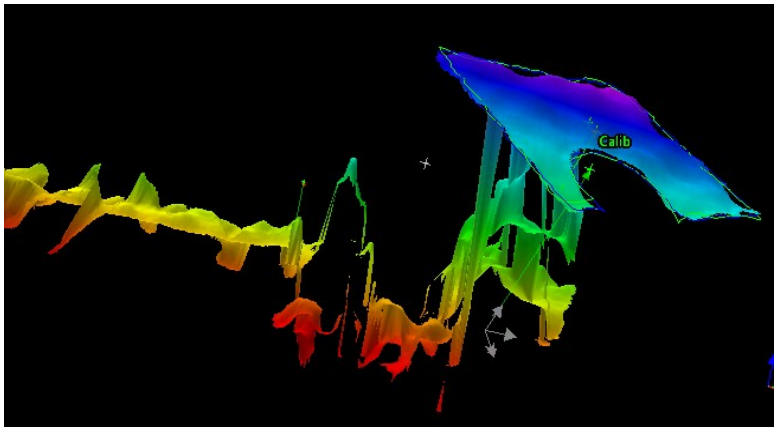
Overlay



If activated in the final 3D Robot Calibration command, a lot of visual information is drawn in the overlay. Also the calibration is applied to the point cloud in the current IM of the command. So the latter should contain a raw camera image.

The red box with its origin cross-hair shows the camera origin pose. This is the point cloud origin before applying the calibration.

The green disc with its origin cross-hair shows the robot base origin pose. This is the point cloud origin after applying the calibration.



For each robot hand pose used for calibration, a gray cross hair is shown at the very center of the robot hand.

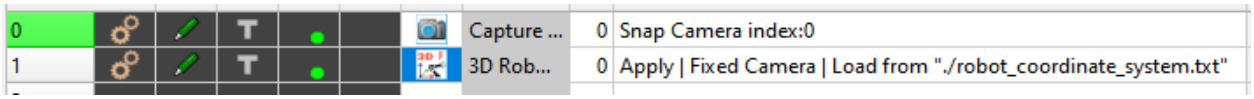
The green point next to it is the object center as predicted by the robot hand position and hand-object offset (green line).

The red point is the object center as measured by the camera. It should ideally overlap with the green point.

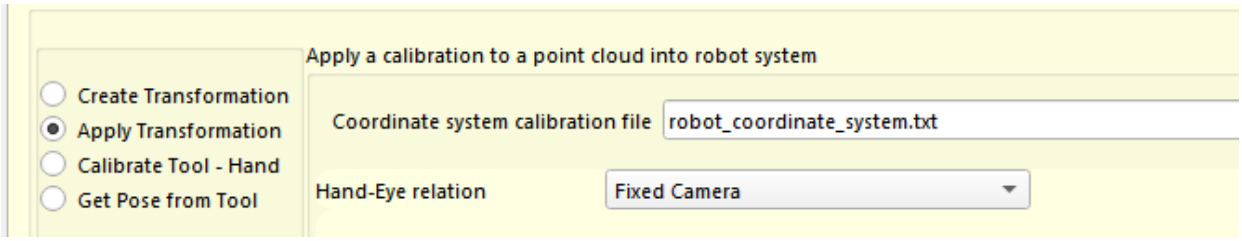
Use this to verify that the calibration is reasonable.

4.5 Calibration Application

Each capture Image command, which reads new point clouds from a file or the camera, needs to be followed directly by a 3D Robot Calibration command:

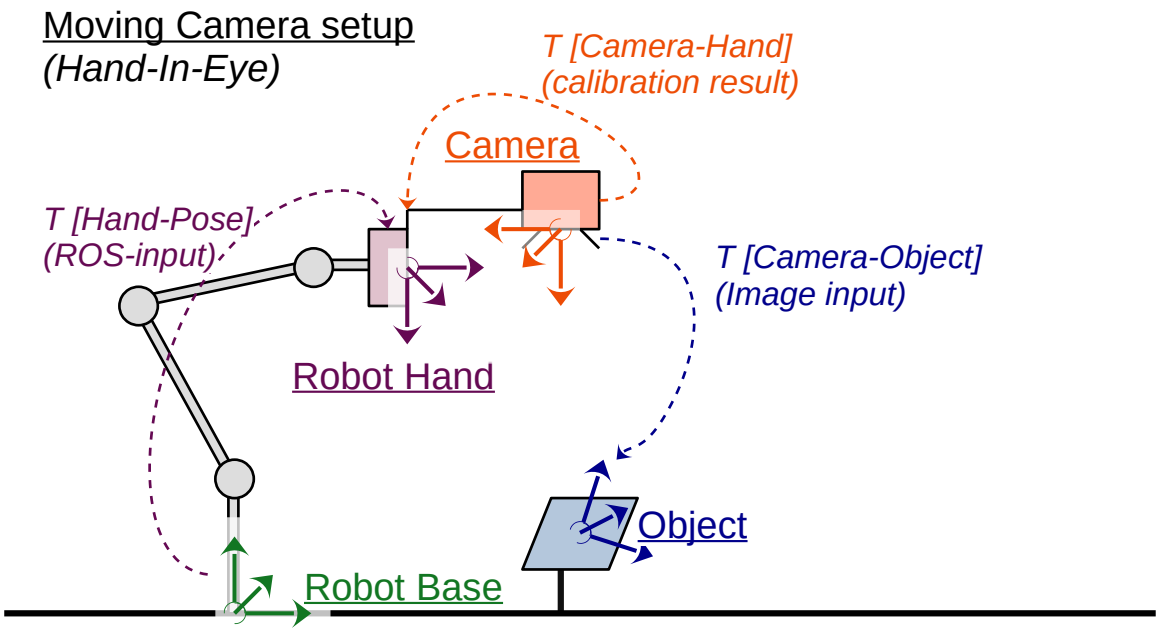


The latter must have the following settings:



After execution of this 3D Robot Calibration command, the origin of the point cloud is set to the robot base. This means, that all positions and poses (rotation-translations) measured in EyeVision are valid target poses/positions for the robot hand.

5 Moving Camera to Robot calibration (Hand-in-Eye)



This section covers the calibration process of a moving camera setup. This means that the camera is not fixed somewhere in the room but is attached to the robot hand. This setup is sometimes referred to as the *Hand-In-Eye* case.

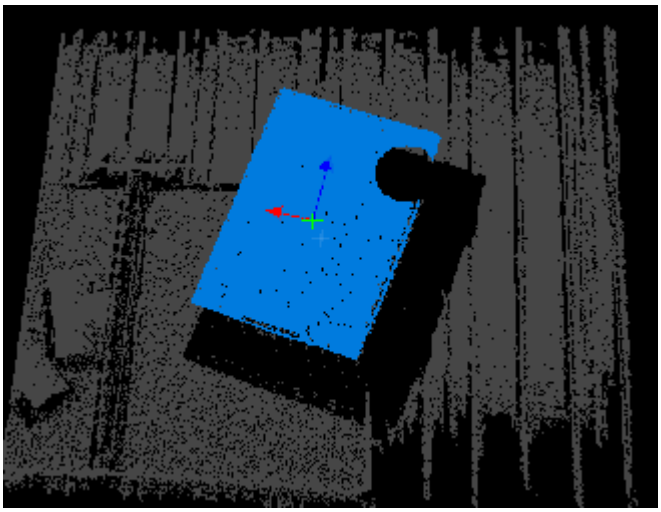
The calibration procedure is analog to the Fixed-Camera setup, but has some import differences. Again this section provides a detailed walk-through with all steps explained.

Again, an object pose is captured from many robot hand poses. The object poses and robot hand poses are input to the 3D Robot Calibration command, which provides the camera to hand transformation as calibration output.

5.0 Calibration Object

The calibration object stays fixed in space. For instance laid down in front of the robot base.

Unlike in the Fixed Camera setup, full pose estimation, this means rotation and position, of the object needs to be detectable with 3D Tools inside EyeVision. This probably excludes the use of balls, cubes, or other objects without a unique pose information.



One example would be to take a punctured box, as shown in the image. Its unique contour provides a reliant 6DoF pose estimation.

A second, more advanced option would be to make use of color images to cut out sections of a surface (Color Filter + 3D Region Map commands) and thus create a contour that is sensitive to rotations.

A third, but also very advanced option would be to use multiple objects together, for instance 3 boxes. This works, since their collective pose can be estimated using the 3D Structure Match command.

5.1 Robot Hand pose and Object pose data

Again, a set of robot hand poses is needed. The bare minimum is 3, but more is recommended. Each robot hand pose must allow the camera, which is fixed to the hand, to see the object.

For more information and the rest of the preparation steps, please follow the instructions in the last section for the Fixed Camera setup, as described in the subsections **Robot Hand calibration poses**, **Capturing calibration images**, and **Finding object pose from images**.

However unlike in the other Fixed Camera setup, it is not sufficient to save the Position of the calibration object. It is crucial that the Rotation-Translation of the calibration object is saved for each pose.

5.2 Final calibration

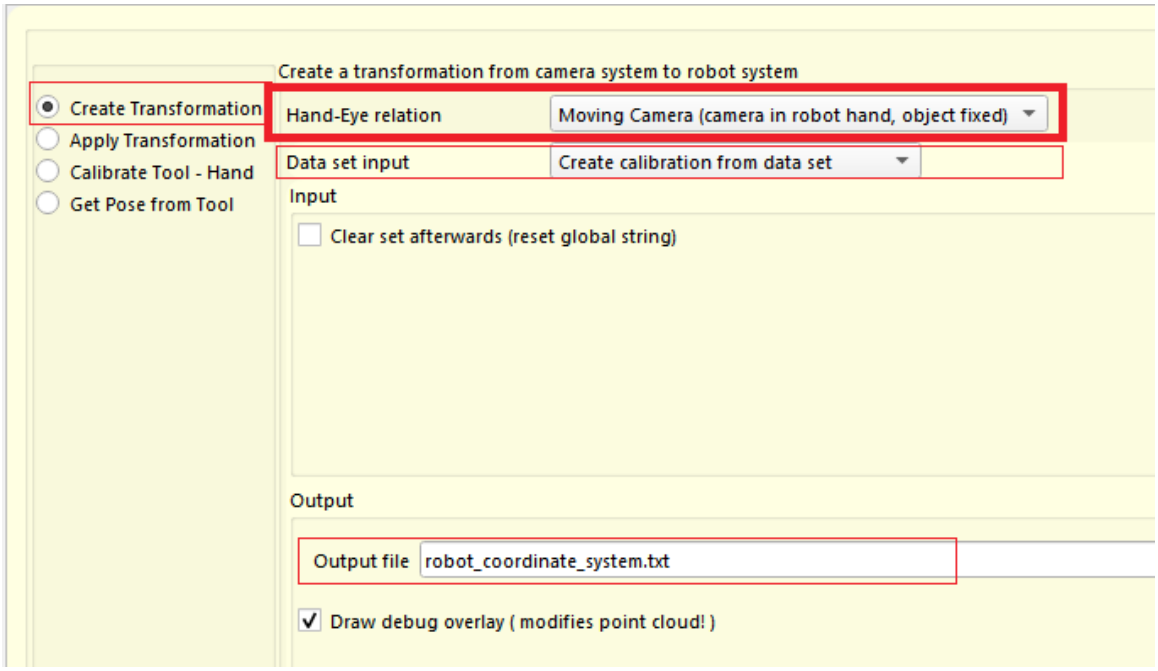
In this step the Robot Hand pose data and Camera-to-Object pose data from the last two sections are used together to calibration the camera to the robot hand relation.

Again it is advised to create an individual program.

Assembling the program

Please see the **Final calibration** subsection in the last chapter and follow all steps accordingly.

The only difference occurs in the parameter settings of the last 3D Robot Calibration command. All parameters should be the same, except for the *Hand-Eye-relation* option, which must bet set to *Moving Camera*!



Results and Output

After now running the whole program once, the value register contains again the mean alignment error. Just like in the Fixed camera case, it is the mean distance in mm between the proposed object position in camera and robot frame. It should be as low as possible. If it is higher than the desired robot hand positioning precision, the calibration process must be revisited.

The Transformation (Rotation-Translation) from camera origin to robot hand frame is written to the *Output file*. This is the *calibration file* which will be used later to apply the calibration. Note that this is an entirely different data than in the Fixed Camera case.

Again, the Geo3D Register (and pickup list) also receive two Rotation-Translation objects as a result of the 3D Robot Calibration command. But the meaning of these rotation-translations is entirely different than in the Fixed Camera setup.

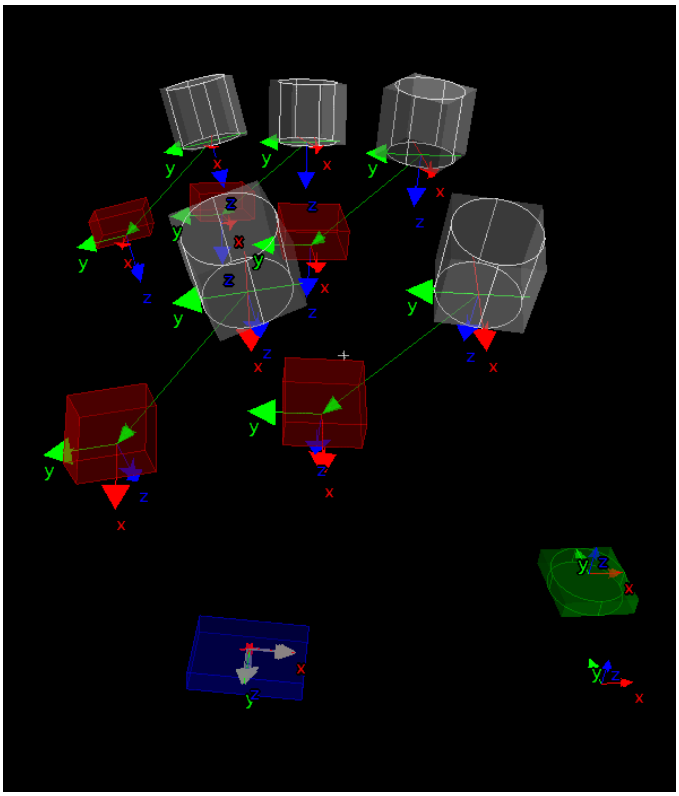
Geo3D register viewer

	Type	Field	x / α	y / β	z / γ	Field	x / α	y / β	z / γ
0	RotationTranslation	Position	-51.557	-85.047	-49.741	Angle	-0.121	-0.108	0.231
1	RotationTranslation	Position	-405.576	-149.478	68.336	Angle	-179.880	0.655	11.615

At index 0 is the main calibration result, i.e. the transformation from Camera to Robot Hand coordinates. This is shown in the sketch at the beginning of this section (orange arrow). This also the data, which is saved to the calibration file.

At index 1 is the 6DoF calibration object pose in the robot base frame. This data is not necessary for future application of the calibration. It only serves as a means to cross-check if the calibration has returned reasonable results.

Overlay

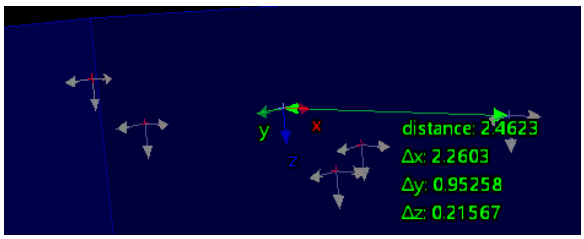


Again, an excessive overlay is drawn into the Image-Memory of the last 3D Robot Calibration command in the program. But unfortunately all point cloud data is removed beforehand, leaving only overlay objects without any camera data.

The green disk shows the origin of the robot base coordinate system, as after applying a calibration.

The blue flat box shows the calibration object. When zooming in the 6DoF pose of the object is shown with the colored cross hair. The predicted pose of the object by the camera images is shown as red dot with gray cross hair:

For each robot hand pose, a gray cylinder with colored cross hair is shown. The red boxes with their cross hairs show the camera origin poses that correspond to the robot hand poses. A connection between these two is shown with the green lines.



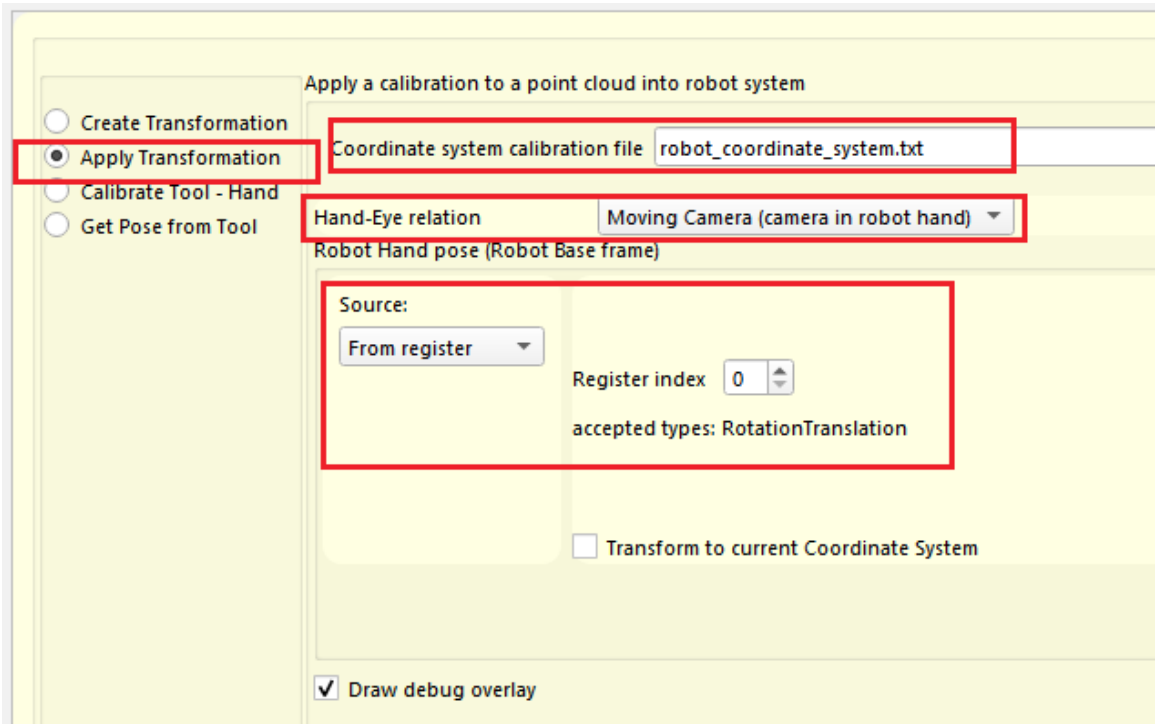
Technical side note: The current IM contains a mock point cloud with very few, artificial points. Each overlay cross hair also has a point in the underlying point cloud. This allows for using the *Measure* tool in the camera viewer as well as *Auto Center* option.

5.3 Calibration application

Applying a calibration to an image from the moving camera again transforms the point cloud origin from the camera to the robot base.

Just like in the Fixed Camera setup, in the Moving Camera setup each Capture Image command must be followed by an instance of a 3D Robot Calibration command:

But this time it is required for each calibration application to know the current robot hand pose, which corresponds to the image taken. The commands parameter settings thus look like the following:



Again, select the calibration file which was created in the last subsection.

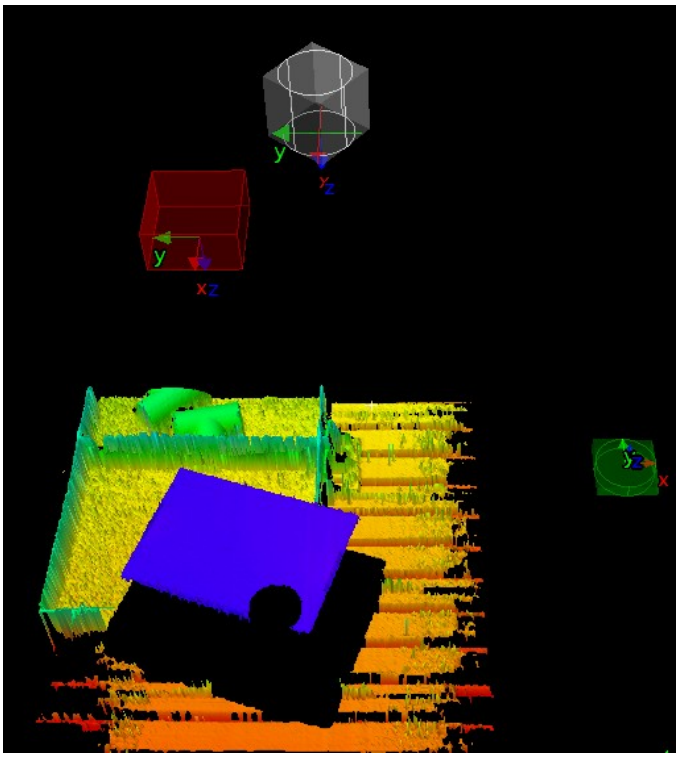
Crucial difference here to the Fixed Camera setup is selecting the *Moving Camera* as *Hand-Eye-relation.* This pops up the input of a current Robot Hand pose, which corresponds the camera image in the current IM.

If selected, some overlay information is drawn into the image:

Green box is the robot base origin, which is now the point cloud origin after applying the calibration.

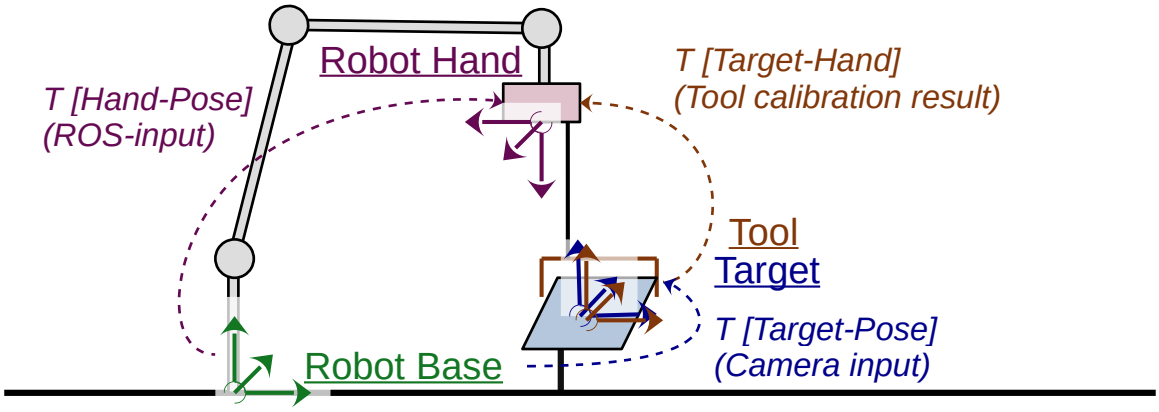
The red box illustrates the camera. Its cross hair shows the former point cloud origin before applying the calibration.

The gray cylinder shows the robot hand pose.



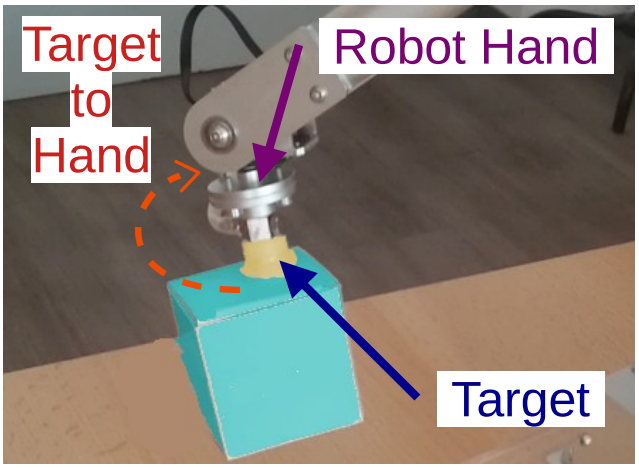
6 Tool calibration

Tool Calibration setup



Having a perfect transformation of the point cloud data from Camera to Robot Base coordinate system, which is the essential goal of the previous chapters, allows to detect all operational targets in the scene in Robot Base coordinates.

Target positions can then be sent as movement destination to the Robot Hand, which is also operated relative to the Robot Base frame.



However, it is often not intended, that the robot hand center will be positioned at the exact same pose as the detected target. Attached tools or other physical anomalies of the robot hand demand, that it is positioned in a certain position and rotation relative to the target. See the image for an example of a vacuum pump, which needs some ~3cm offset in Z direction from the Target. Target describes in this case the upper surface of the box, since this is the object which EyeVision will probably detect.

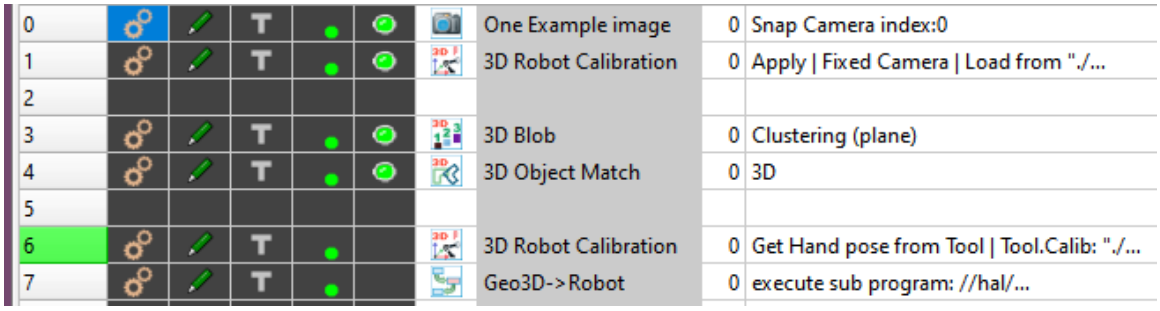
The key data which is needed for correct hand pose calculations is the 6DoF pose of the robot hand in the target frame. i.e. the Rotation-Translation from target object pose to robot hand pose.

In some generic cases, this can be estimated by the user. But in more complicated cases, like highly rotation sensitive tools, tilted tools, unknown hand center positions, a reliant user estimation of the Target-to-Hand offset is not possible.

However, this is why the next sections will first describe how to apply Target-to-Hand offsets for finding Robot Hand poses based on scene target results. And afterwards it will be explained how to acquire the Target-to-Hand offset data.

6.0 Getting Robot Hand pose from Target

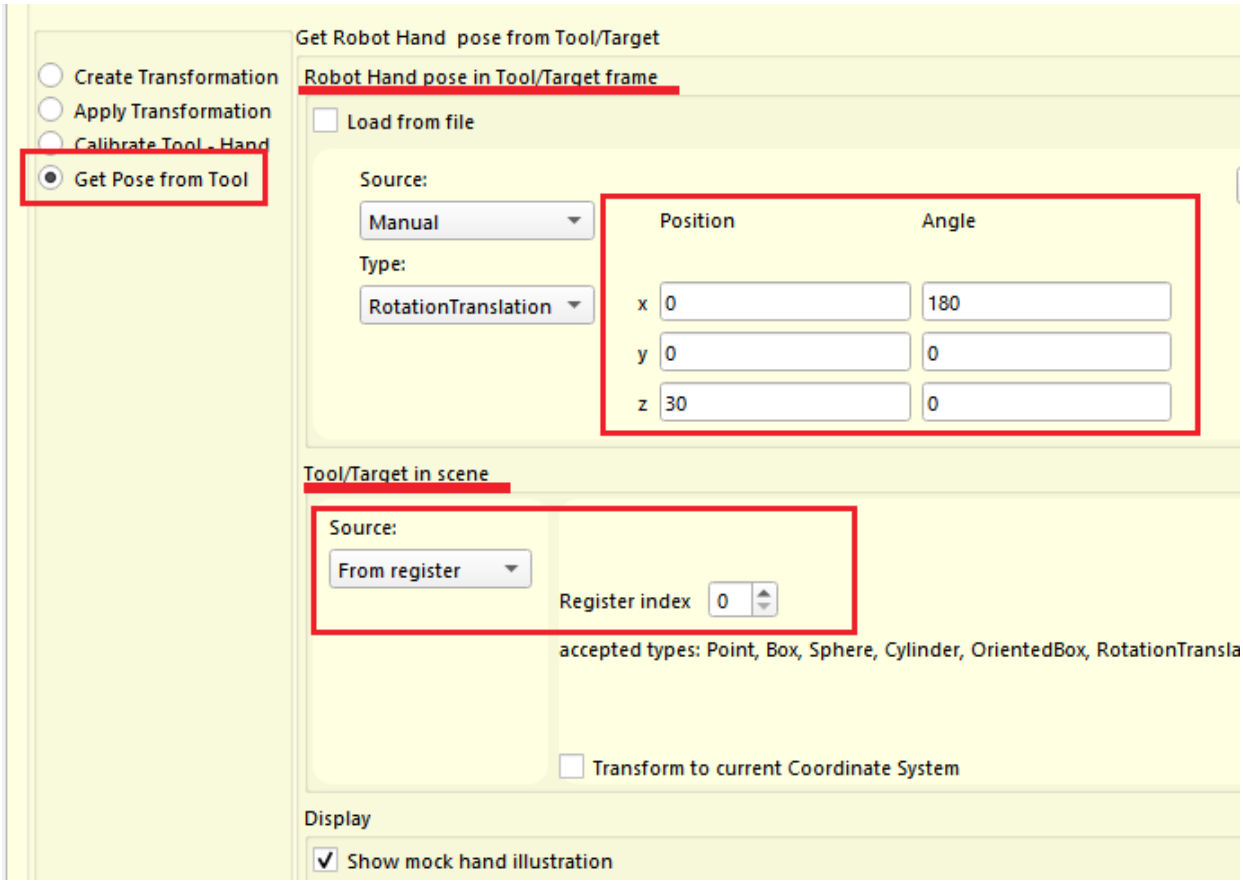
A typical application program for bin picking could look like this:



After capturing a point cloud (line 0), transforming it to robot base (line 1), and detecting desired target objects (line 3, 4), the Register and Pickup-List contains one or more Rotation-Translation objects of the targets objects.

Now a 3D Robot Calibration command at line 6 can be used to calculate the Robot Hand pose, which is needed to correctly approach the target with respect to the tool offsets.

The settings of this command would look like this:



The *Robot Hand pose in Target frame* is the Target-to-Hand offset as discussed above.

It can be either provided manually or with a file, containing the Rotation-Translation data.

A guide how to find these offsets will be given in the next two sections for manual and automated procedure respectively.

The *Target in scene* is the object pose as found by the 3D commands. It can be read from the register or pick-up list.

Note that if you provide no rotation information, e.g. by providing Point as input, the rotation will be taken as 0. This may work in some cases but needs extra caution, since it supposes that orientation is never important.

Geo3D register viewer

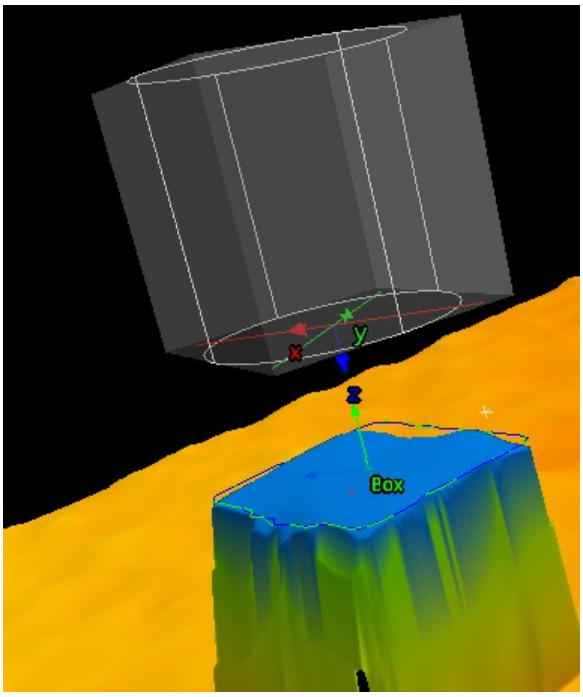
	Type	Field	x / α	y / β	z / γ	Field	x / α	y / β	z / γ	Field
0	RotationTranslation	Position	38.497	243.160	42.994	Angle	-176.045	0.664	-21.253	
1	RotationTranslation	Position	38.844	245.229	13.068	Angle	3.955	-0.664	21.253	
2										

Executing the command will put a Rotation-Translation object into the register and the pickup-list. In the image below the input target pose is at index 1. The resulting robot hand pose is at index 0.

Now the data at index 0 can be sent to the robot as destination for the robot hand pose. If done so, the tool will eventually touch the target object.

The 3D Robot Calibration command also creates overlay information, if selected. A gray cylinder with colored cross hair illustrates the robot hand center.

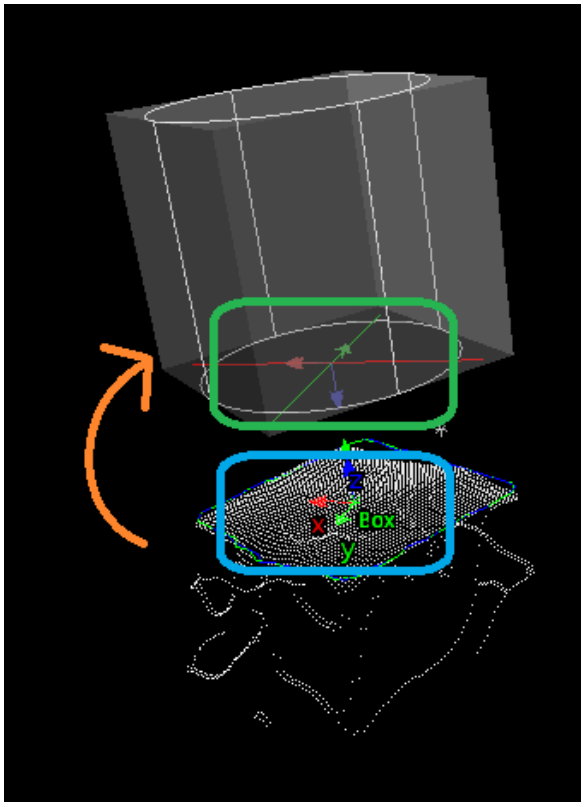
Please see that the robot hand is transformed from the object according to the input in the command parameter interface.



6.1 Manual estimation of Target-to-Hand offset

Some cases allow for a manual estimation of the Target-to-Hand transformation by the user. This is for instance the case if the object must be grabbed with a vacuum pump.

First, the rotation from object to hand must be found. For this it is crucial to understand the orientation of the objects in the scene as detected by EyeVision.



In the example on the right, a flip of the Z direction is necessary. The objects Z direction is pointing upwards, whereas the robot hand needs to come down from above.

The easiest way to achieve this is one 180° rotation around the X axis. Additional rotations around the Z axis can also be added without harm.

The position values are the position of the hand center in the object frame. In the example with the vacuum pump, the hand center needs to be placed along the vertical axis of the object, right above the object center. So X, Y position values can be left at 0.

The Z value is simple the positive distance between vacuum pump end and robot hand center. This could be roughly estimated as 30mm in this example. If trying it out shows that there is still a gap between pump and object, this value must be increased. If the pump pushes into the object, the Z value must be decreased.

6.2 Automated Target-to-Hand calibration

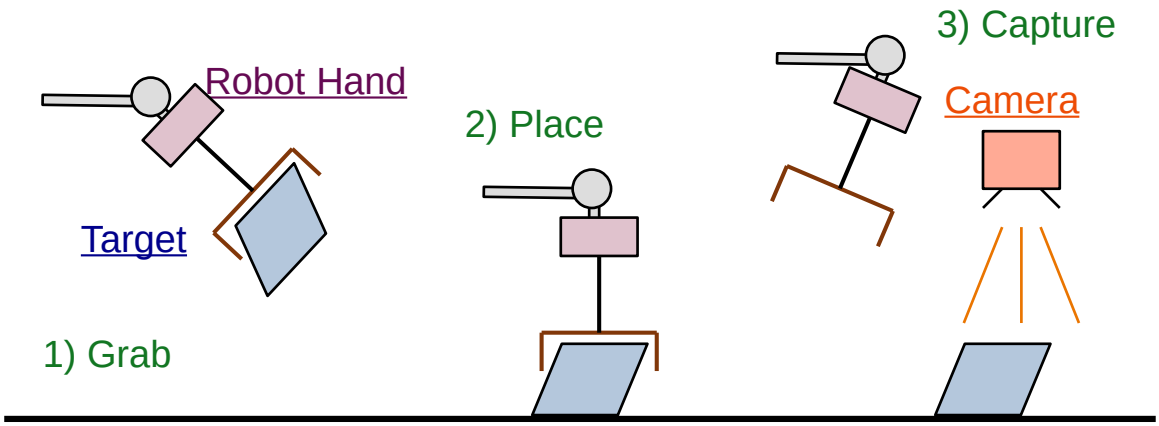
The 3D Robot Calibration command also allows to automatically calculated the tools Target-To-Hand offset.

For this two inputs poses are needed: The first is a pose of an example object in the scene as captured by the camera (and calibrated to robot base coordinates). The second is the Robot Hand pose in an ideal grabbing position for the object.

In the following two different approaches are illustrated of how to get this data.

Pose acquisition via Placing

Tool Calibration Steps *(Drop-Mode)*



1) Let the tool (e.g. gripper) grab the object.

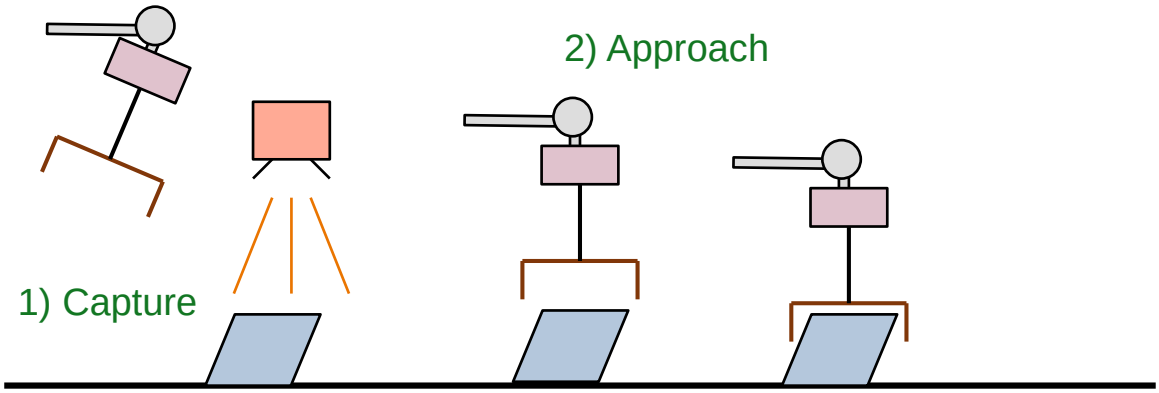
2) Move the Robot Hand down to the ground. Place the object gently such that it does not fall down or move when being released.

X) Read/Save the robot pose to file or 3D Register.

3) Move the Robot Hand out of the Image. Capture and save an image.

Pose acquisition via Picking

Tool Calibration Steps *(Pick-Mode)*



1) First, place an example object under the camera. Capture and save an image.

2) Move the Robot Hand manually to the object such that the tool reaches ideal grabbing pose.

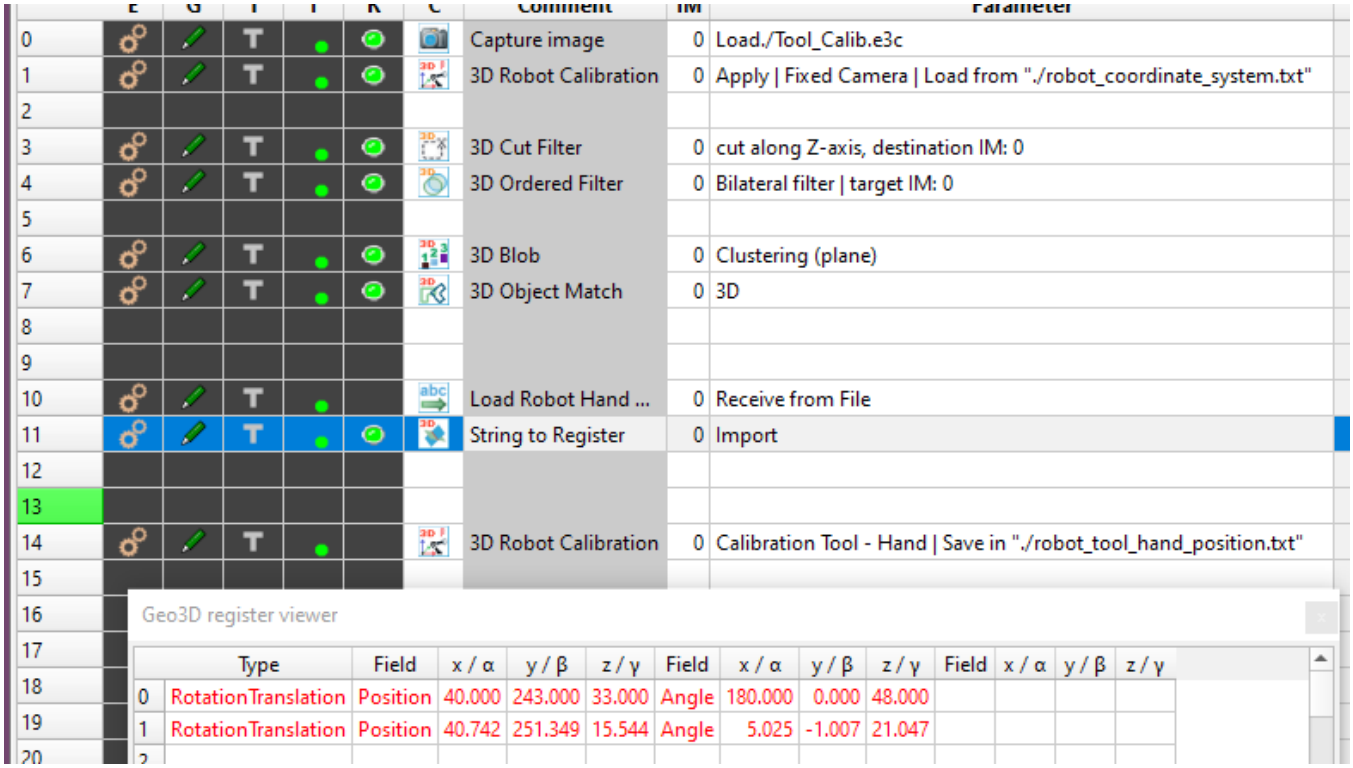
X) Read/Save the robot pose to file or 3D Register.

Final Tool Calibration

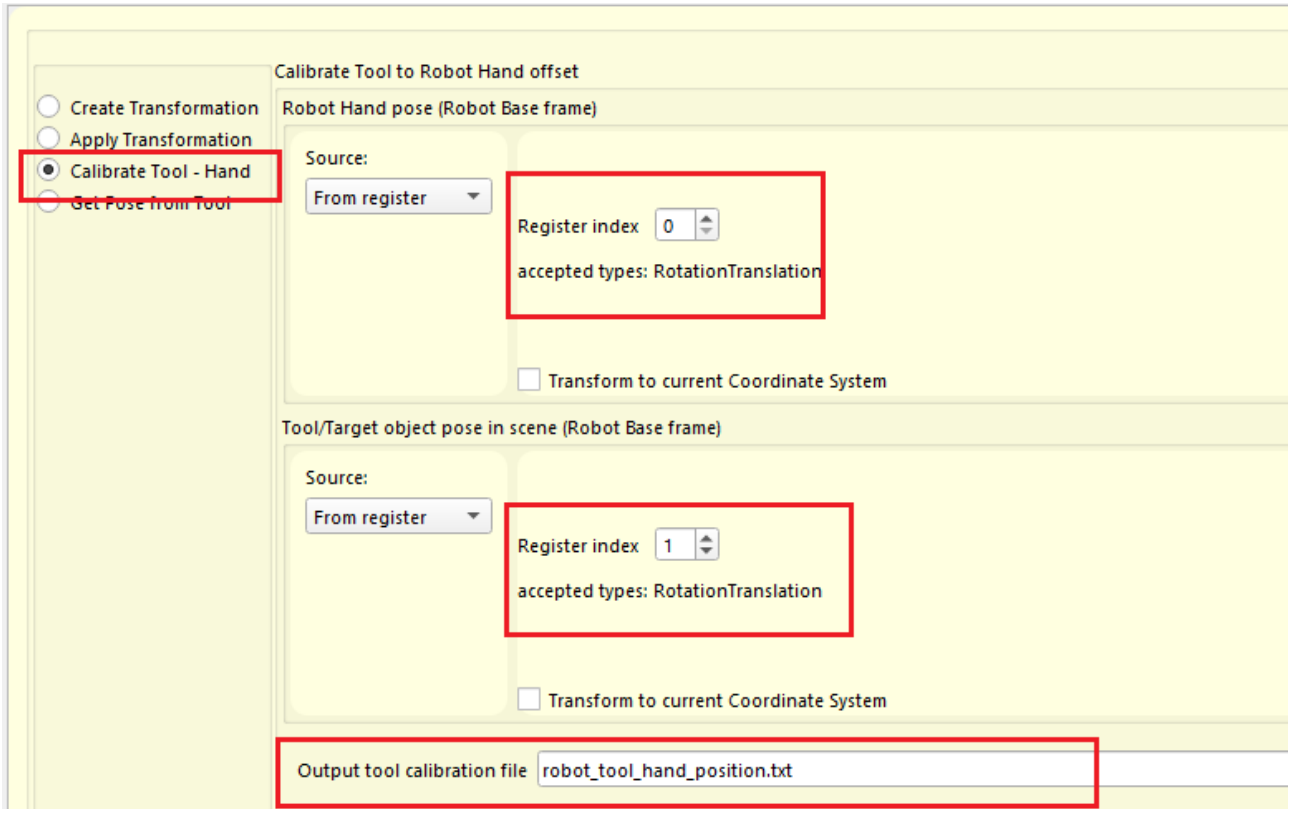
Now in an EyeVision program two things must happen:

First, the target object pose must be found. For this, use the desired, regular tool chain for object detection as it is to be used later. This should put the object pose into the 3D Register. In this example 3D Object Match puts the box pose in the Register at Index 1.

Second, the robot hand pose is needed. This can be entered manually or read from a file. In this example it is shown in the register at index 0.

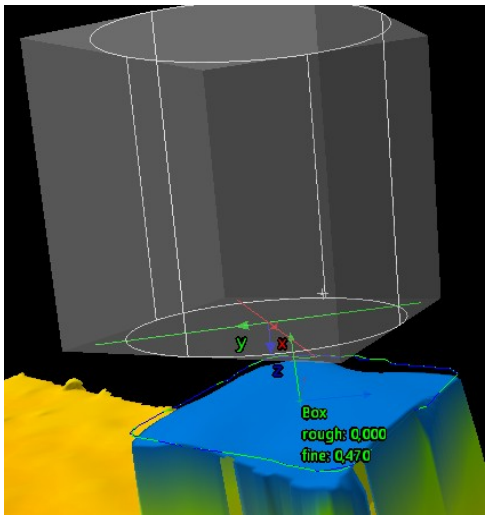


No a 3D Robot Calibration command is needed with the following settings:



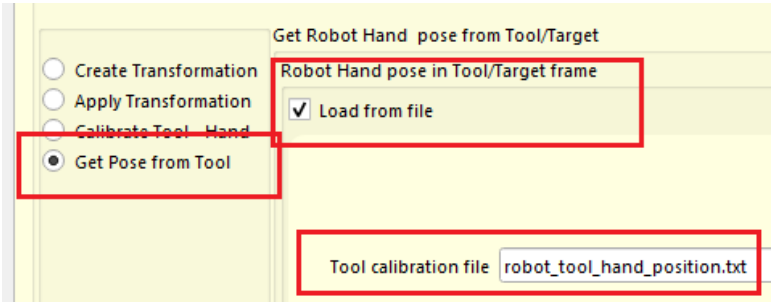
The indices need to be adjusted according to the source and order of the pose inputs.

After execution, the command writes Target-to-Hand, (aka "*Robot Hand pose in Tool/Target frame*") to the file specified.



An overlay is also drawn, showing the Robot Hand illustration as defined in the input.

Tool Calibration file, which is created by the command, now can be used as an input for the *Get Pose from Tool* operation to replace the manual input:



7 Application example

4__Application.ckp

	E	G	T	I	R	C	Comment	IM	Parameter
0							ROBOT OUT OF PICTURE		
1		0)					Home position	0	create RotationTranslation from simple types
2							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
3									
4							IMAGE IN ROBOT COORDS		
5			T				Capture image	0	Snap Camera index:0
6			T				3D Robot Calibration	0	Apply | Fixed Camera | Load from "./robot_coordinate_system.txt"
7									
8			T				3D Cut Filter	0	cut along Z-axis, destination IM: 0
9			T				3D Ordered Filter	0	Bilateral filter | target IM: 0
10									
11							FIND ALL OBJETS		
12			T				3D Blob	0	Clustering (plane)
13			T				3D Object Match	0	3D
14									
15			T				Interpreter control	0	Refresh image
16									
17							LOOP ALL OBJECTS		
18			T				3D Loop	0	start loop over 3d ObjectList
19									
20							Above Object		
21		A)					3D Robot Calibration	0	Get Hand pose from Tool | Tool.Calib: from variable
22							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
23									
24							On Object		
25		B)					3D Robot Calibration	0	Get Hand pose from Tool | Tool.Calib: "./robot_tool_hand_position.txt"
26							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
27									
28			T				Activate Grabber	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
29									
30							Above Object		
31		A)					3D Robot Calibration	0	Get Hand pose from Tool | Tool.Calib: from variable
32							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
33									
34							Above Dropboff		
35		C)					3D Robot Calibration	0	Get Hand pose from Tool | Tool.Calib: from variable
36							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
37									
38							At Dropboff		
39		D)					3D Robot Calibration	0	Get Hand pose from Tool | Tool.Calib: from variable
40							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
41									
42			T				Release Grabber	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
43									
44							Above Dropboff		
45		C)					3D Robot Calibration	0	Get Hand pose from Tool | Tool.Calib: from variable
46							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...
47									
48			T				3D Loop	0	end loop
49									
50		0)					Home position	0	create RotationTranslation from simple types
51							Geo3D->Robot	0	execute sub program: //hal/eyevision_projects/public/3D_Area_Examples/...

The screenshot above shows an example for an entire bin picking application with a robot arm.

Walk through

Geo3D->Robot is a the name of a sub-program which is called multiple times during this program. In here the communication with the robot is done in which the latest RotationTranslation at the top of the Geo3D Register is send to the robot.

Line 1,2 Moving the robot Hand to a fixed position, which is not visible by the camera.

Line 5,6 Capturing a point cloud. Calibration into Robot Base coordinates.

Line 6-13 Anaysis toolchain: Cut working area, filter surface, blob surfaces, match box template.

Line 18,48 Loop over all objects in the object list.

Line 21,22 Position the robot hand above the object with some distance. The target-to-hand offset pose is given manually in the 3D Robot Calibration command, since it is no precise operation. (Pose A)

Line 25,26 Position the robot hand exactly at the object such that the grabber touches it. The precise target-to-hand offset was calibrated earlier. (Pose B)

Line 28 Activate the grabbing unit at the robot hand.

Line 30,31 Move back to the previous pose. (Pose A)

Line 35,36 Move above the drop-off position. (Pose C)

Line 39,40 Move at the exact drop-off position. (Pose D)

Line 42 Deactivate the grabbing unit to drop the object.

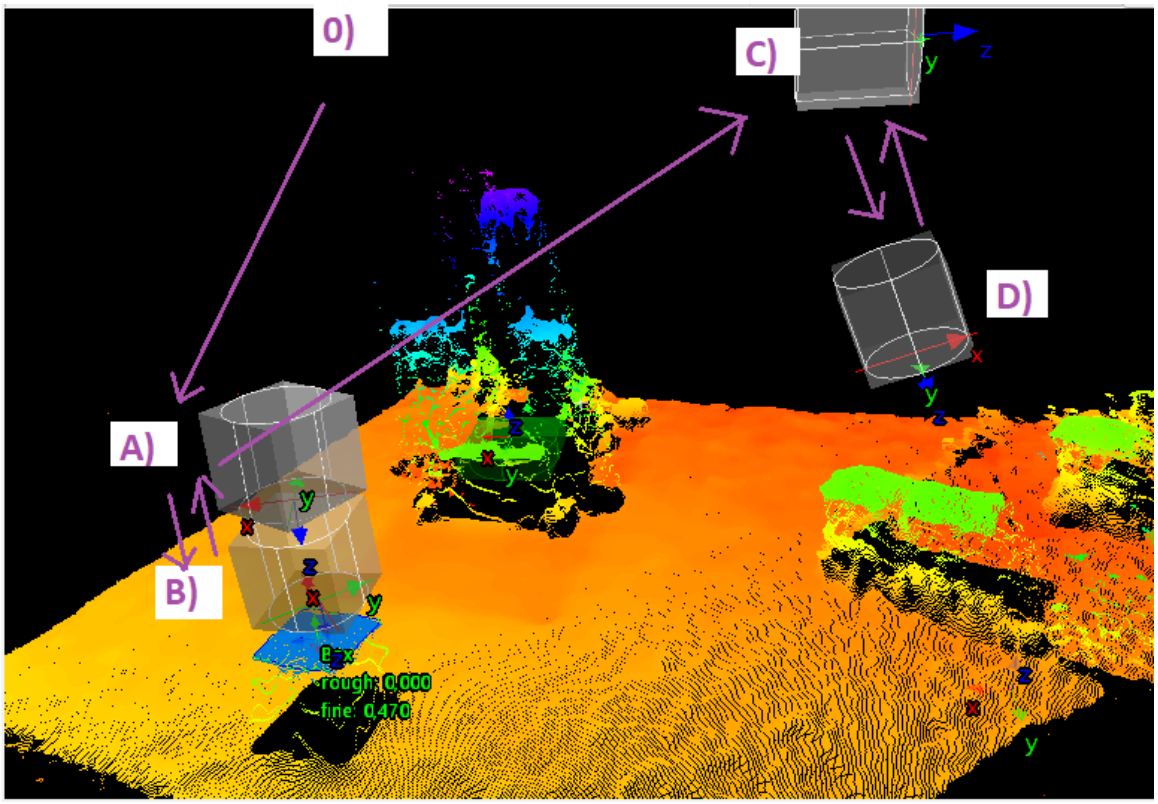
Line 45,46 Move back above the drop-off position. (Pose C)

Line 50,51 At the end move back to the parking pose out of the cameras view.

The ImageMemory 0, in which the point cloud, objects, and the overlay is drawn, could look like the following screenshot.

Here the purple arrows illustrate the movement of the robot hand for picking and placing one box object. At each pose, the gray cylinder with colored cross-hair shows the exact hand pose.

The green disk with the cross-hair in the back of the image is the point cloud origin in the robot base center.


0)
C)
D)
A)
B)
rough: 0.000
fine: 0.470

8 Troubleshooting / FAQ

No common problems/questions known yet.

Will be added on demand.

9 Euler Angle rotation notations

All 3D rotation variables in EyeVision (*Rotation*, *RotationTranslation*, *OrientedBox*) are defined with three angles in the Z→Y→X extrinsic Euler angle convention.

The robot operation systems (ROS) however often use different rotation convention than EyeVision. This means that a value conversion into/from EyeVision must take place during the 6DoF Pose communication from EyeVision to the robot and in the reverse case as well.

The functionality for this is built into the Robot Communication command for easy use. For further information about the usage of the command please read the context help of the command inside EyeVision.

For more general information about Euler angle rotation notations, please read the following document:

http://www.evt-web.com/fileadmin/downloads/releases/doc/3d_rotations_in_eyevision.pdf