Mech-Viz Manual

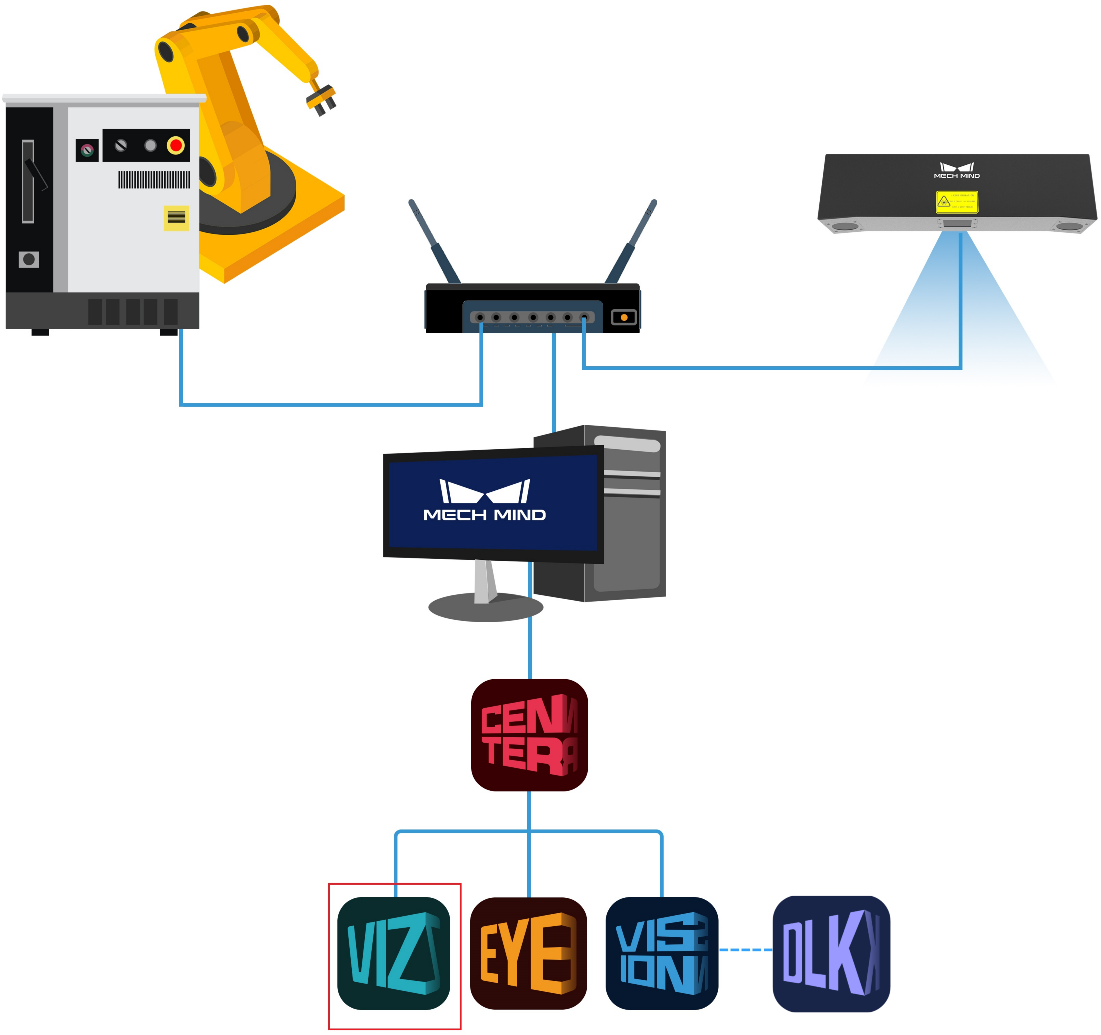
Mech-Mind

Jul 01, 2022

CONTENTS

Introduction

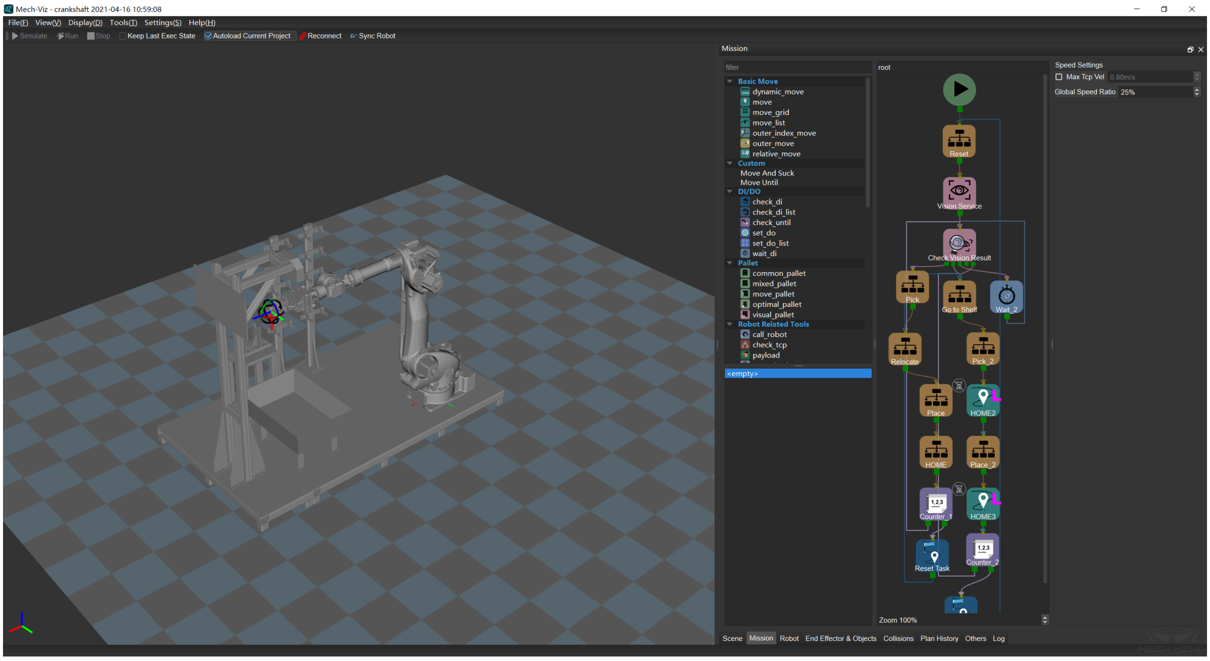
Mech-Viz is a **graphical programming environment** software developed by Mech-Mind to control industrial robots. Providing a control and simulation environment in the robot control system, Mech-Viz enables users to visualize the simulation and control robots by graphical programming in cooperation with Mech-Vision, Mech-Center, etc., as shown in the figure below.



Mech-Viz has a user-friendly interface as shown in the figure below.

It has a variety of independently-developed and high-efficiency tools for developing visual solutions.

With a combination of advanced graphical programming model and visual simulation module, developers who do not know any robot programming language can program and control a robot in a simple and direct way, which greatly reduces the time and cost required by robot programming.



Mech-Viz offers an unified interface for robots of various models and brands. Mech-Viz is also equipped with several built-in intelligent algorithms for decision-making, autonomous path planning, program inspection, etc., which can be seamlessly integrated with Mech-Mind Software Suite.

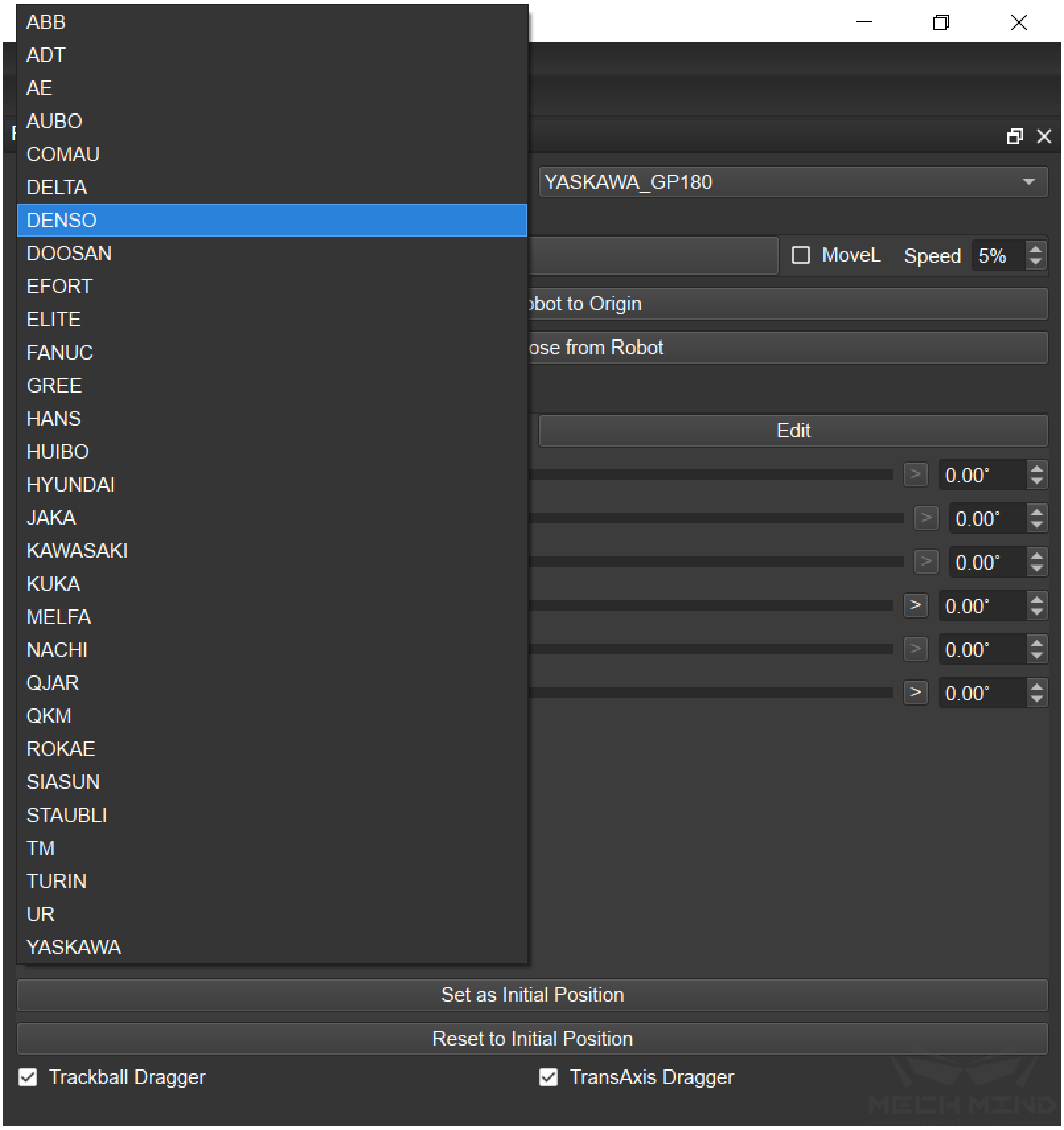
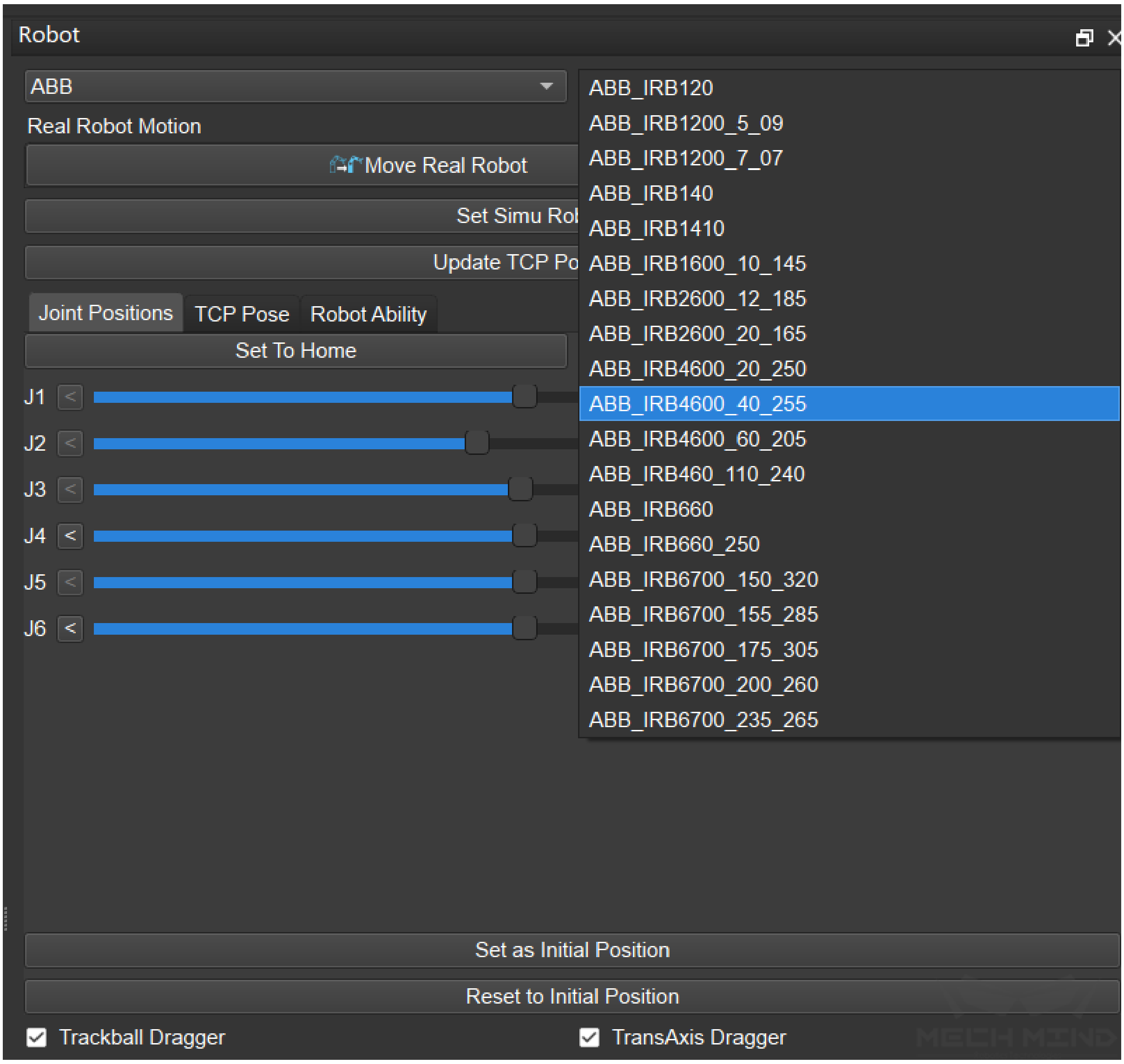


ABB
ADT
AE
AUBO
COMAU
DELTA
DENSO
DOOSAN
EFORT
ELITE
FANUC
GREE
HANS
HUIBO
HYUNDAI
JAKA
KAWASAKI
KUKA
MELFA
NACHI
QJAR
QKM
ROKAE
SIASUN
STAUBLI
TM
TURIN
UR
YASKAWA
YASKAWA_GP180
□ MoveL
Speed 5%
obot to Origin
ose from Robot
Edit
0.00°
0.00°
0.00°
0.00°
0.00°
0.00°
Set as Initial Position
Reset to Initial Position
☑ Trackball Dragger
☑ TransAxis Dragger


Robot
ABB
Real Robot Motion
Move Real Robot
Set Simu Rol
Update TCP Po
Joint Positions
TCP Pose
Robot Ability
Set To Home
J1
J2
J3
J4
J5
J6
ABB_IRB120
ABB_IRB1200_5_09
ABB_IRB1200_7_07
ABB_IRB140
ABB_IRB1410
ABB_IRB1600_10_145
ABB_IRB2600_12_185
ABB_IRB2600_20_165
ABB_IRB4600_20_250
ABB_IRB4600_40_255
ABB_IRB4600_60_205
ABB_IRB460_110_240
ABB_IRB660
ABB_IRB660_250
ABB_IRB6700_150_320
ABB_IRB6700_155_285
ABB_IRB6700_175_305
ABB_IRB6700_200_260
ABB_IRB6700_235_265
Set as Initial Position
Reset to Initial Position
☑ Trackball Dragger
☑ TransAxis Dragger

CHAPTER

ONE

FUNDAMENTALS

1.1 Robot

Unless otherwise specified, robot in this article refers to a system of rigid bodies connected by joints, as shown in *Figure 1*.

Figure 1. Example of robot

1.2 TCP (Tool Center Point)

Robots are usually equipped with end effectors, which are used to interact with objects in the surrounding world. The tool center point (TCP) is the tip point of the end effector. In order to complete tasks such as picking, we usually say that the robot should move to a specific point in space, which actually means its TCP should move to that point.

1.3 Joint Position (Jps)

Joint position, commonly known as joint angle, is the position of each joint of the robot. Joint positions are a set of parameters, and the number of sets of parameters equals the number of joint axes. For example, the set of joint positions of a six-axis robot is 6.

1.4 Pose

The position and orientation of objects in space are collectively called poses. The pose is expressed in translation and rotation (quaternion or Euler angle). In Mech-Viz, poses are divided into tool pose and object pose, as shown in *Figure 2*.

1.4.1 TCP pose

The pose of TCP relative to the base of the robot is the TCP Pose.

1.4.2 Object Pose (Obj Pose)

The object pose is the pose of the object’ s center relative to the base of the robot. When an object is attached to the end of the robot, the default initial position of the object is the same as the tool pose, but the orientation is opposite. If there is *Symmetry* or *Pick Tool Offset* , the relative transformation of the object pose and tool pose may change.

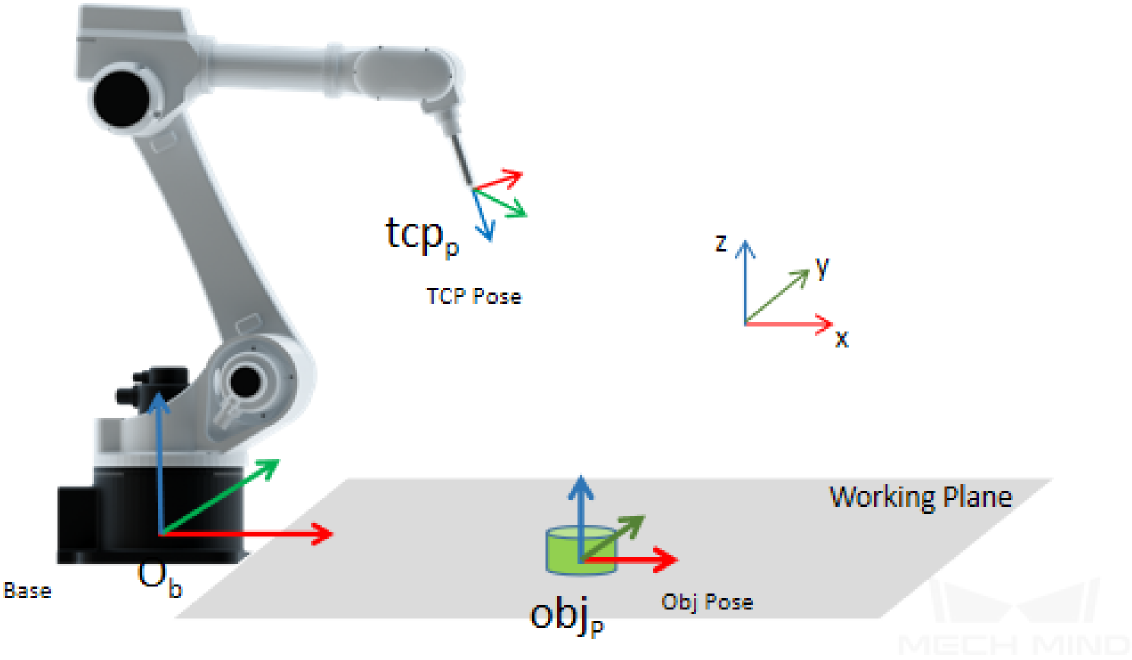


Figure 2. Poses

Attention: The direction of the Z-axis of the object pose must be the opposite of the direction of the Z-axis of the TCP pose during the grasping process.

1.5 Forward Kinematics

Forward Kinematics computes the pose of the end effector from given joint angles. When a set of robot joint angles is given, there is only one solution of the tool pose.

1.6 Inverse Kinematics

Inverse Kinematics computes the joint angles from a given end effector pose. For a specific tool pose, the corresponding joint angles may have multiple sets of solutions or no solutions.

1.7 Pick-Hold-Place

Picking, holding and placing describe the way that robot interacts with objects. These actions are an integral part of applications of symmetry in *Movement Tasks* and transformation changes/resets between object poses and tool poses.

1.7.1 Picking

The state that robot moves to the pick point pose and controls the end effector to pick the object by changing the output signal is called “picking”. Once the “picking”state becomes effective, the relative position of the object and the robot’s TCP is bonded, and Mech-Viz will plan the robot’s trajectory according to the object’s *Symmetry*.

1.7.2 Holding

The state after the robot picking up the object and before the object being placed is called “holding”. In the “holding”state, the bound relationship between the object and the robot always exists. This state will take effect automatically after the object is picked, and cannot be configured in Mech-Viz.

1.7.3 Placing

The state that robot reaches the pose of placing and controls its end effector to release object by changing the output signal is called “placing”. When the “placing”state becomes effective, the relative position of the object and the robot’s TCP is no longer bonded.

1.8 Symmetry

Each object to be picked has a corresponding object pose. According to the object pose, Mech-Viz will change the tool pose to control the robot to pick, as described in *Pose*. In practice, objects often have symmetry. For these objects, the robot can pick or place in various ways according to its **symmetry angle**, and the results are the same.

> **Symmetry angle** is the smallest angle for which the object can be rotated around the X/Y/Z axis to coincide with itself. For example: the symmetry angle of a square is 90°; of a rectangle is 180°; of a regular hexagon is 60°; of a circle or a ring is 0°; no rotation symmetry is 360°, as shown in *Figure 3*.

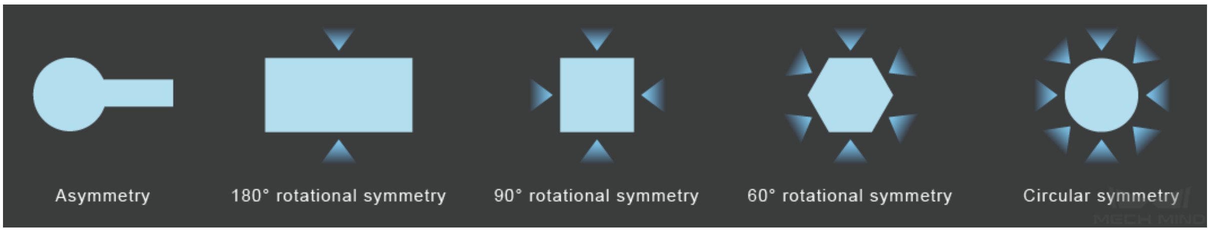


Figure 3. Symmetry angle of objects

Among them, the symmetry around the Z axis is called the strong axis symmetry, and the symmetry around the X or Y axis is called the **weak axis symmetry** .

> **weak axis symmetry** : Usually, between X and Y axis, the axis with a relative stronger symmetry will be regarded as the weak axis and its symmetry will be used in the actual calculation. For example, if an object has 90° symmetry angle around X axis and 0° symmetry angle around Y axis, the Y axis will be regarded as the weak axis, and therefore the symmetry angle of weak axis is 0°.

Users can set the symmetry of the object and use Mech-Viz to select an optimal trajectory of picking and placing to improve accessibility and reduce the rotation of the end effector.

1.8.1 Symmetry of Object

Object' s symmetry is a necessity for the adjustment of object' s pose. For example, when the symmetry of the object is 180°, the robot can freely pick or place in either "forward" or "backward" direction to reduce the rotation of the end effector, as shown in *Figure 4*.

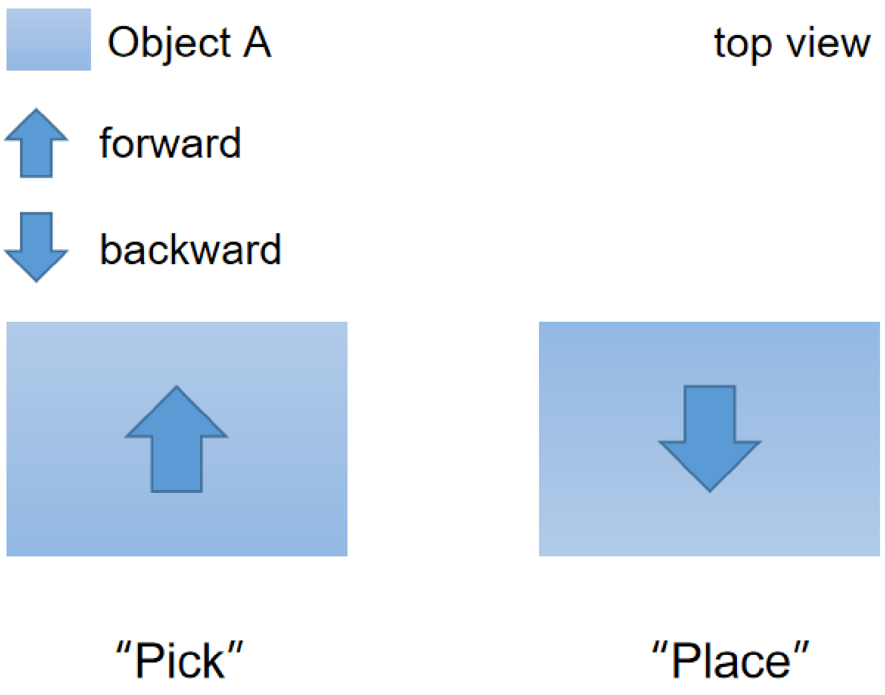


Figure 4. Object Symmetry

The two states of the object can be regarded as the same, and robot can pick in the forward direction and place in the backward direction.

1.8.2 Symmetry of Pick Point

The symmetry of the pick point is related to the way that the end effector picks the object and is used to select an optimal pose of picking. For example, when the object is smaller than the size of a suction cup, the object is picked by the vacuum gripper with offset in order to prevent collision with objects nearby during the picking process. In this case, the object symmetry does not take effect, and the pick point has a symmetrical angle of 180°. You can rotate the pick tool at the initial pick point to "forward pick" or "backward pick", as shown in *Figure 5* below:

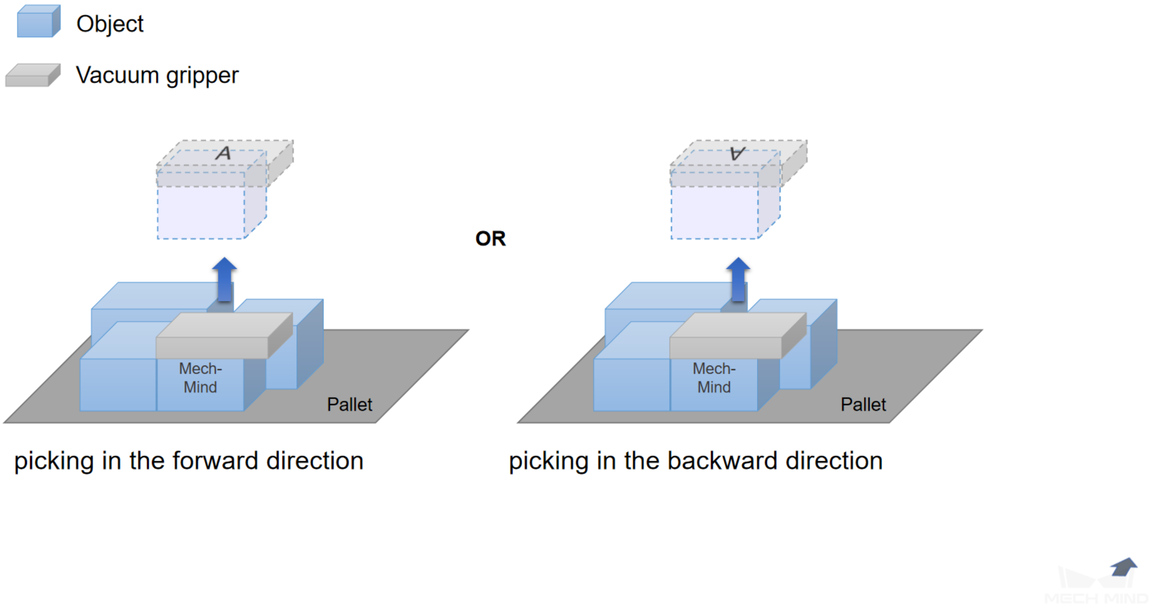

Object
Vacuum gripper
A
OR
Mech-Mind
Pallet
picking in the forward direction
picking in the backward direction

Figure 5. Forward and backward pick

In order to plan the optimal trajectory of movement, Mech-Viz will consider both the symmetry of the pick point and the object during the picking process; and the object pose will be selected according to the symmetry of the object during “holding” and “placing” . For example: when the object is a connecting rod, you can use an adaptive gripper to pick the ring from inside out, as shown in *Figure 6* below:

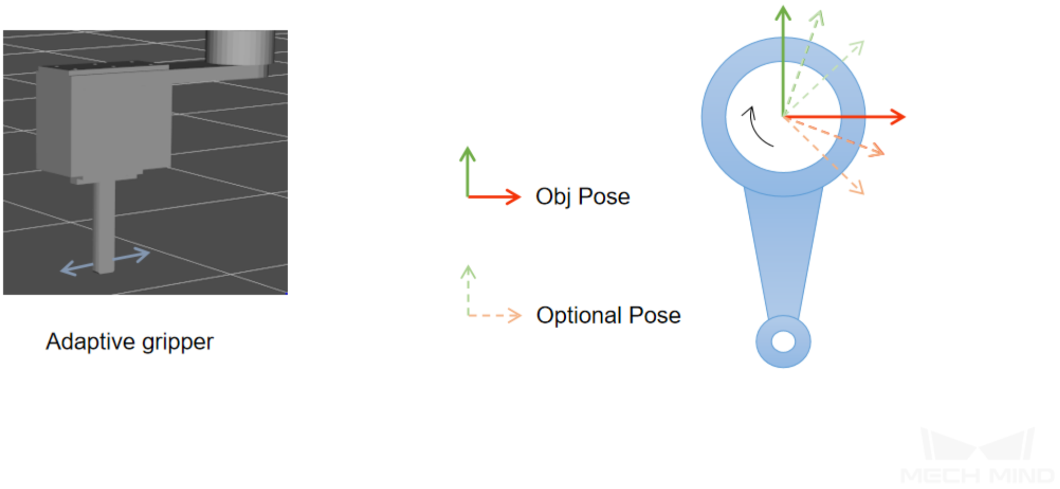


Figure 6. Pick a ring from inside out

The symmetry of the pick point here is 0°, and any angle is optional during picking. Therefore, the symmetry of the pick point should be set with a reasonable trial step size (such as 1°, 10°, etc.). However, since there is no object’ s symmetry, if you want to place the object in a set position, Mech-Viz will restore the relative transformation between the end effector and the object during picking.

1.9 Pick Tool Offset

When the visual result guides the robot to pick objects of small size or more complex shapes, in order to avoid collision with other objects, the TCP can be offset to a certain point of the object to pick. After assigning the type of tool and strategy of picking, Mech-Viz calculates the value of bias and determines with the point cloud collision to check whether it can be picked or not. As shown in *Figure 7*.

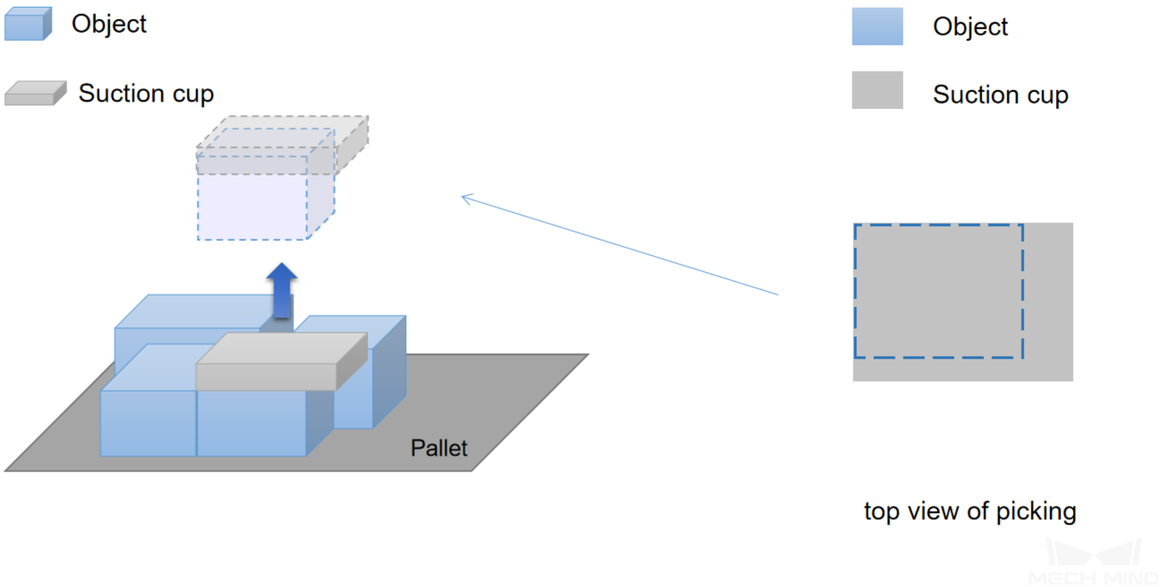


Figure 7. Pick tool offset

Similarly, in mix palletizing tasks, robot can avoid collision with objects nearby with the usage of pick tool offset .

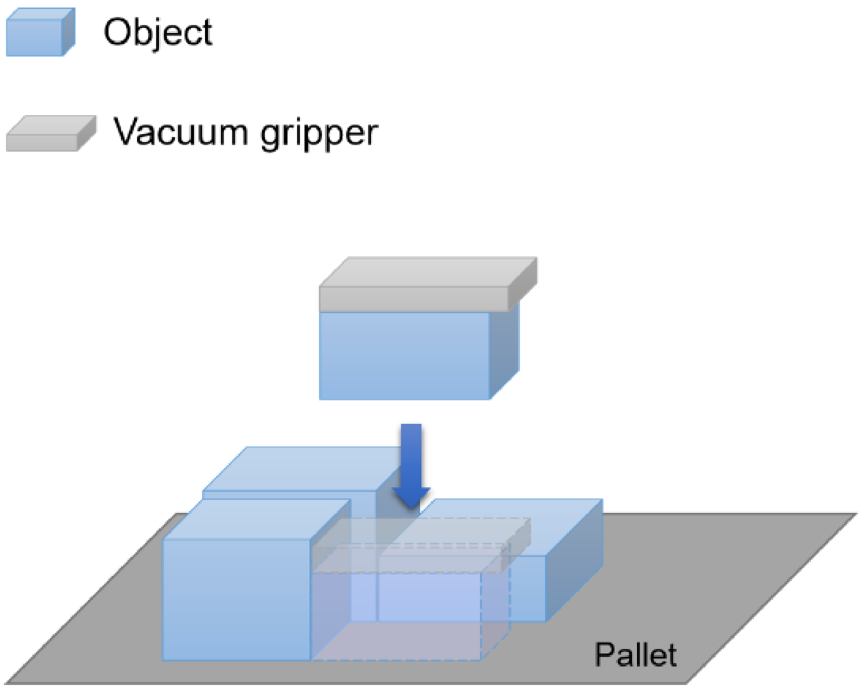


Figure 8. Pick tool offset to avoid collision

Note: For more detail about the using of bias, please see *visual_move*.

1.10 Collision Model Type

There are 4 main types of collision models in Mech-Viz:

1. Basic geometry (only include cuboid at present): This type of model mainly includes cuboid and end effector which is only loaded with a 3D model but without a collision model in the scene.
2. Mesh (only include the surface information): This type of model only enables collision detection of the surface. The main models are robot joints and end effector loaded with STL collision model.
3. Convex polyhedron assembly: This type of model is used as a solid for collision detection, and the main model is an end effector loaded with OBJ collision model.
4. Octree: The main models are point cloud and the end effector loaded with binvox collision model.

CHAPTER

TWO

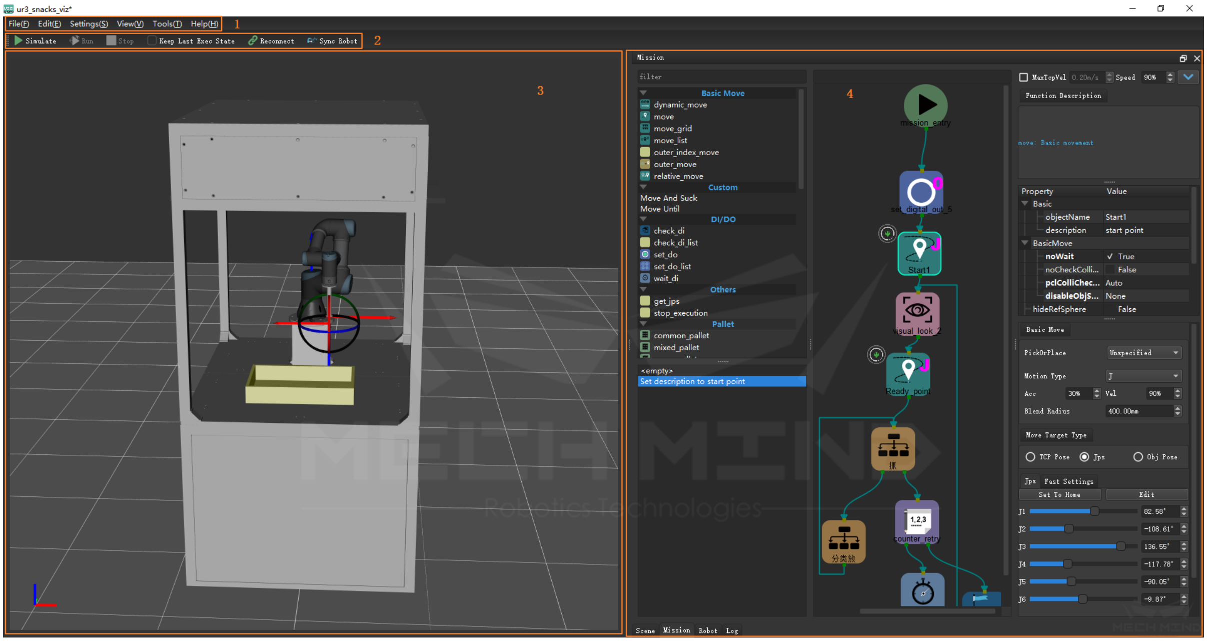
INTERFACE

2.1 User Interface

The following is a brief description of the software from the perspective of its main components and function implementation.

2.1.1 Main interface

Click the desktop icon to open the main interface of the software (as shown in *Figure 1*).



<empty>
Set description to start point

☐ MaxTcpVel 0.20m/s Speed 90%

move: Basic movement

Property	Value
Basic	
objectName	Start1
description	start point
BasicMove	
noWait	✓ True
noCheckColli...	False
pclColliChec...	Auto
disableObjS...	None
hideRefSphere	False

PickOrPlace: Unspecified

Motion Type: J

Acc 30% Vel 90%

Blend Radius 400.00mm

○ TCP Pose ● Jps ○ Obj Pose

J1	82.58°
J2	-108.61°
J3	136.55°
J4	-117.78°
J5	-90.05°
J6	-9.87°

Figure 1 Mech-Viz main UI

The main interface can be divided into

1. Menu Bar
2. Control Toolbar
3. 3D Simulation Area

4. Mission Editing Area

The robot’ s movement simulation, actual controlling and graphical programming are realized through the synergy of each area.

2.1.2 3D Simulation

The 3D simulation area is used to display in real time the running status of the robot in real or simulated execution of program, including the robot’ s movement trajectory, highlighting of collision prediction, cloud point and pick point both given by vision service.

Scene Object Display

As shown in *Figure 2*, the 3D simulation area is mainly composed of three parts:

1. The grid like ground
2. The imported robot
3. The imported or created scene model and end effector model

The mouse wheel can be used to zoom in and out of the entire display area. Left-clicking anywhere on the display area and holding down the left mouse button you can drag the scene to adjust the viewing angle of the display area, and holding down the mouse wheel can shift perspective.

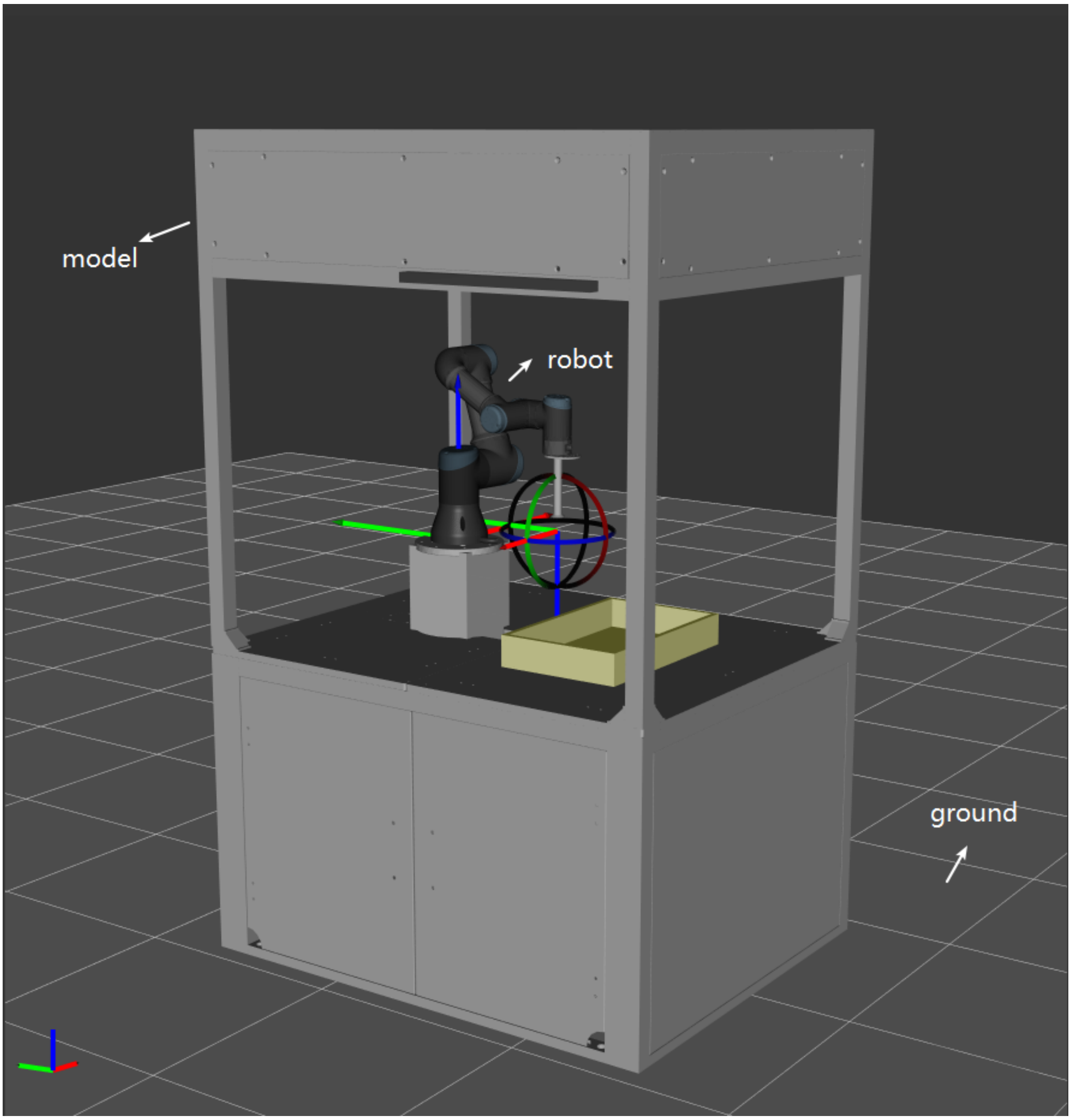


Figure 2 The 3D simulation area

Movement Trajectory Display

Whether in controlling a real robot or in simulation, the 3D simulation area can display the trajectory of the robot. As shown in *Figure 3*, the user can plan the robot’s movement more reasonably through the simulation trajectory.

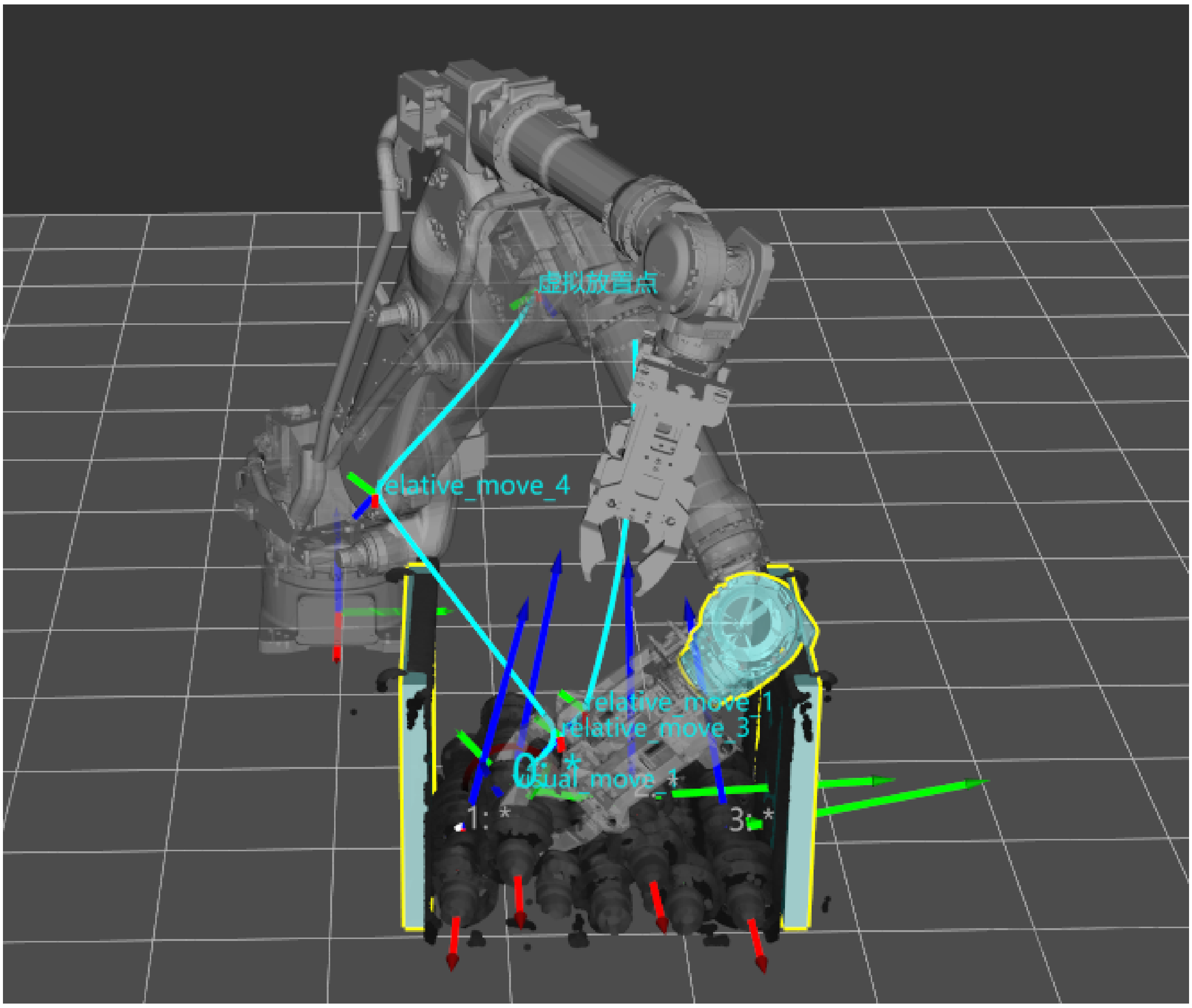


Figure 3 Movement simulation

Collision Prediction Display

As shown in *Figure 3*, during the task running, collision detection will be performed by the software on the scene objects in the planned trajectory and end-effectors. If a collision is predicted, it will be highlighted in the 3D simulation area and a more reasonable trajectory will be selected to control movement of robot.

Point Cloud and Pick Point Display

The 3D simulation area will update the display of object point cloud and pick point every time after calling the visual service and taking a picture. The displayed information include grabbing order, object' s type, pose and so on.

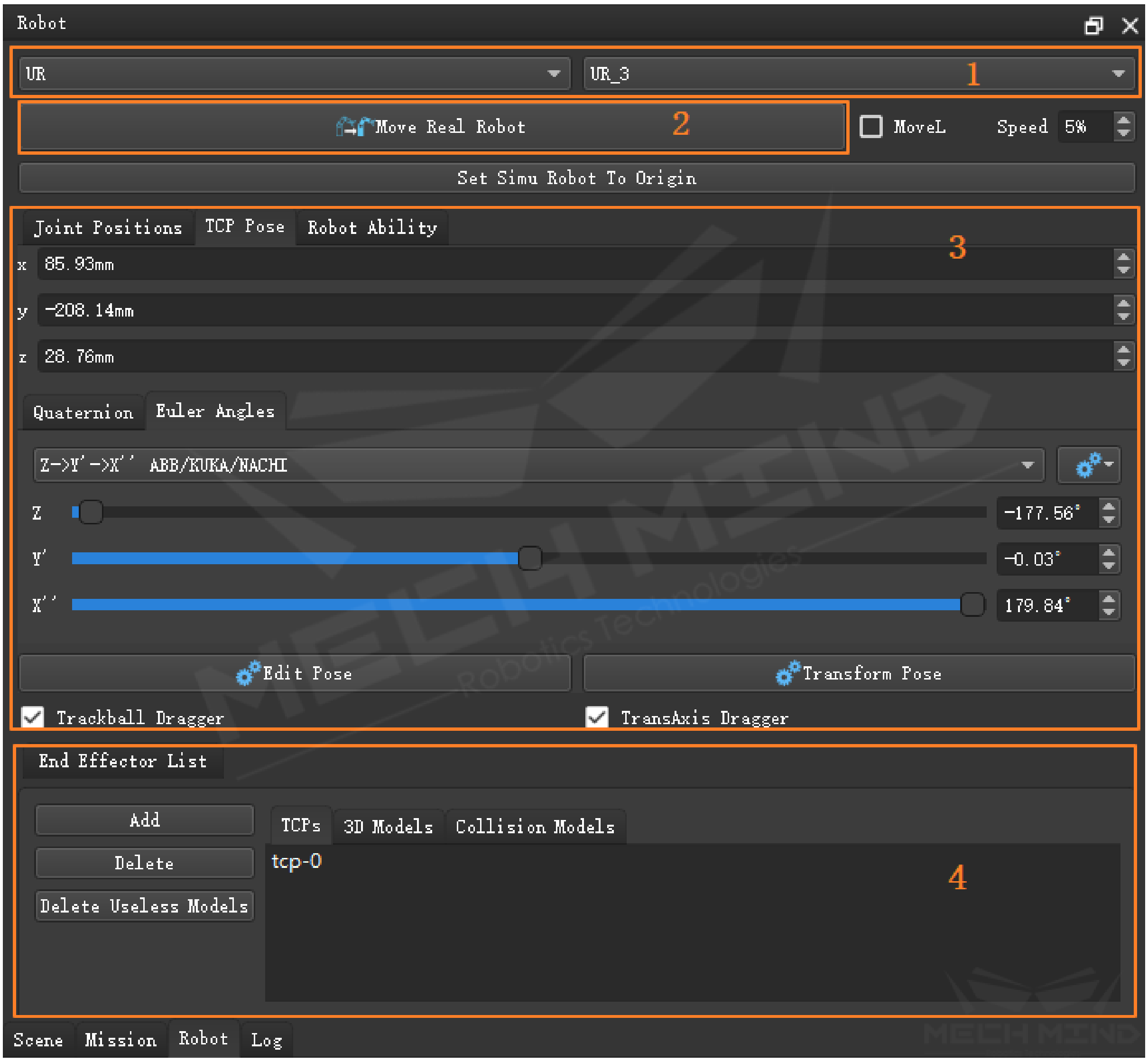
2.1.3 Mission Editing

The mission editing area is divided into four tabs, namely the Scene tab, Mission tab, Robot tab and Log tab, which can be used to complete the establishment of scene objects, robots and end effectors models, create motion programs with graphical programming, and set or adjust global parameters. The four tab pages can be dragged to adjust the order of display or floated in the screen. The functions of each tab are described below.

Robot Tab

When building a project, you usually start by importing a robot. The robot tab is used to create a simulated robot or operate a connected real robot. As shown in *Figure 4*, the functions of the robot tab include:

1. Select a robot. You can select the robot brand and model from the drop-down menu at the top of the page so that the corresponding simulated robot can be created and displayed in the 3D simulation area.
2. Move a real robot synchronously. When the real robot is connected, it can move to the current pose of simulated robot controlled by Mech-Viz.
3. Adjusting the pose of a simulated robot. Whether it is a four-axis or six-axis robot, you can adjust the robot' s pose by changing the joint angle or TCP pose.
4. An end effector model can be imported for display and collision detection, and the center point of TCP coordinate system can be adjusted according to the installation pose of the real end effector.



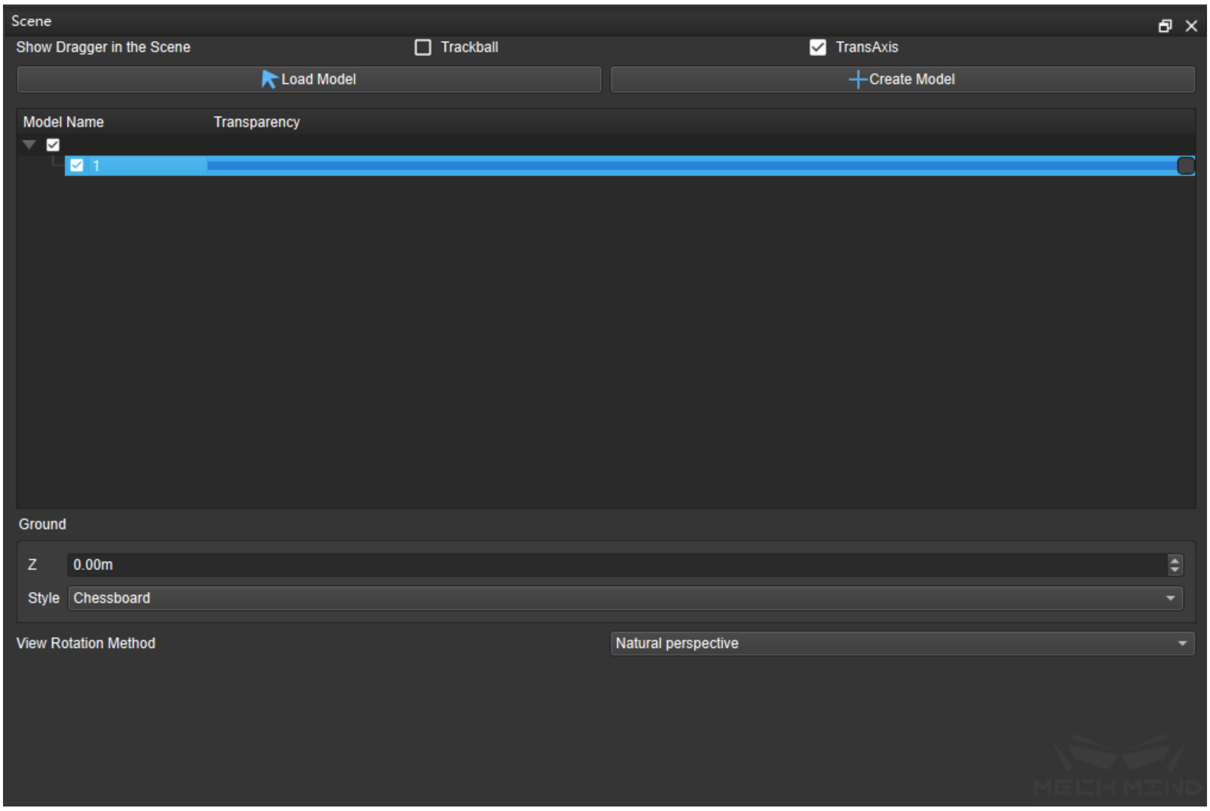
UR
UR_3
□ MoveL
Speed 5%
x 85.93mm
y -208.14mm
z 28.76mm
Z->Y'->X'' ABB/KUKA/NACHI
Z -177.56°
Y' -0.03°
X'' 179.84°
☑ Trackball Dragger
☑ TransAxis Dragger
tcp-0

Figure 4 Parameters in robot tab

Scene Tab

The function of the scene tab is mainly to construct the scene where the entire robot works, including the workbench, workspace, etc. As shown in *Figure 5*,

1. you can import or create the scene object model and adjust transparency of the model to achieve the modeling of the scene.
2. In the “Edit Target Object” option, the symmetry and symmetry range of the target object can be adjusted, which is used to plan the optimal trajectory and the grab point with minimum collision during the process of picking and placing.
3. Ground Grid: By adjusting the height of the scene ground, the relative position relationship of the scene objects can be adjusted.



Show Dragger in the Scene □ Trackball ☑ TransAxis

Z 0.00m

Style Chessboard

View Rotation Method Natural perspective

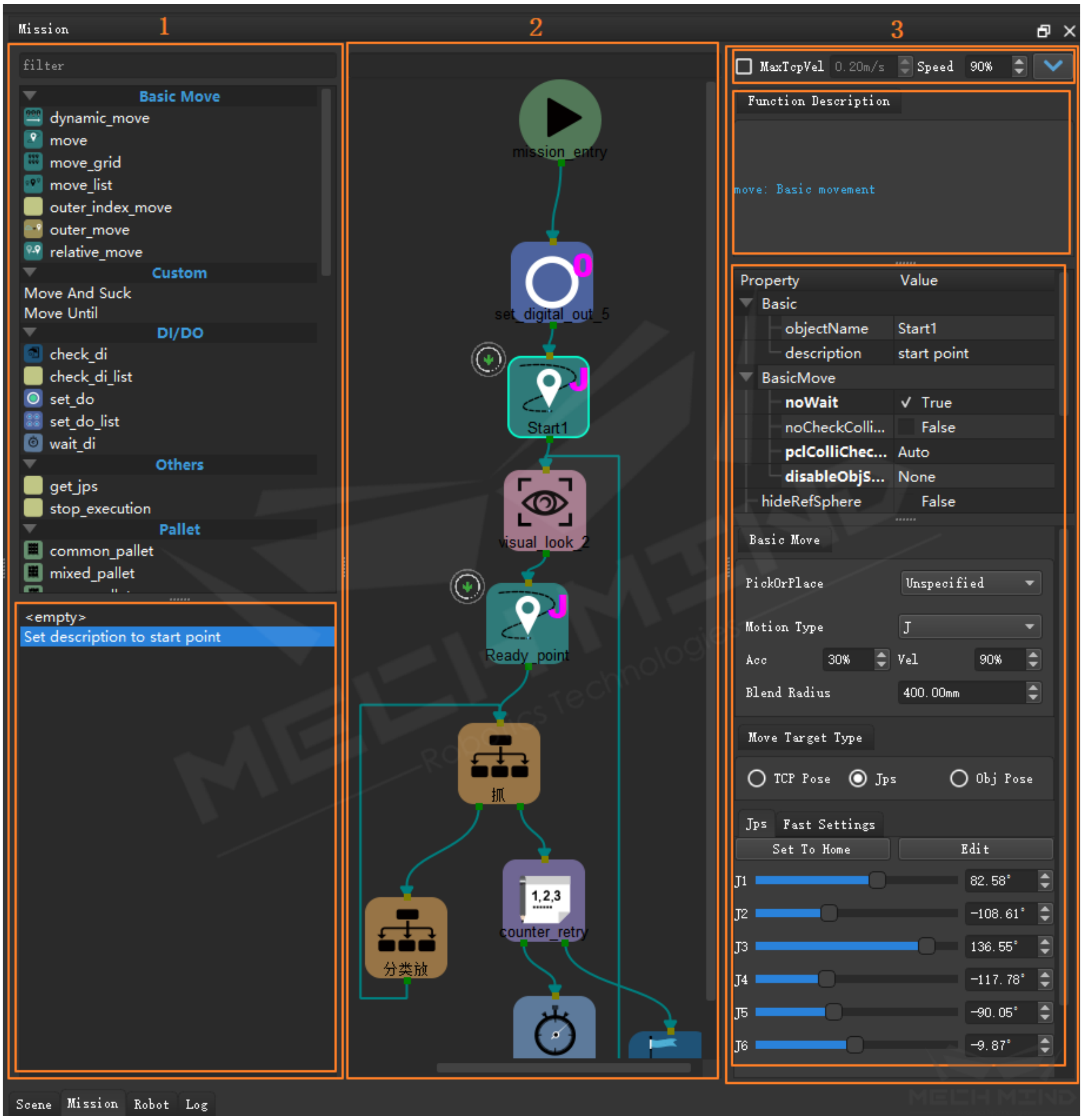
Figure 5 Main functional areas in scene tab

- *Load Model* : you can load the model file in the stl, dae format.
- *Create Model* : you can create a simple model in the Scene directly.

Mission Tabs

You can program for the robot in the mission tab

As shown in *Figure 6*, it consists of module list, graphical programming workspace, and parameter panel.



<empty>
Set description to start point

□ MaxTcpVel 0.20m/s Speed 90%

move: Basic movement

Property	Value
Basic	
objectName	Start1
description	start point
BasicMove	
noWait	✓ True
noCheckColli...	False
pclColliChec...	Auto
disableObjS...	None
hideRefSphere	False

Basic Move

PickOrPlace: Unspecified

Motion Type: J

Acc 30% Vel 90%

Blend Radius 400.00mm

Move Target Type

○ TCP Pose ● Jps ○ Obj Pose

Jps

J1 82.58°
J2 -108.61°
J3 136.55°
J4 -117.78°
J5 -90.05°
J6 -9.87°

Figure 6 Main functional Areas in Mission Tab

1. There are many modules in the module list area, which have variety of functions. Drag a module to the programming area, then set the parameters of this module in parameter panel area, you can drag more modules as needed, after that, connect those modules, a program is complete.

   The operational history will be shown in the lower part of the module list area, which make it more convenient to debug.

2. The programming area is mainly used to arrange the corresponding programming modules and connect them in a certain order to achieve the programming. As shown in *Figure 7*, each programming module has an input port (small brown square) and an output port (green small square). After clicking the output port we can get a connection line and drag the connection line to the

next module input port to implement the connection of programming modules.

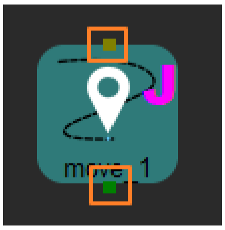
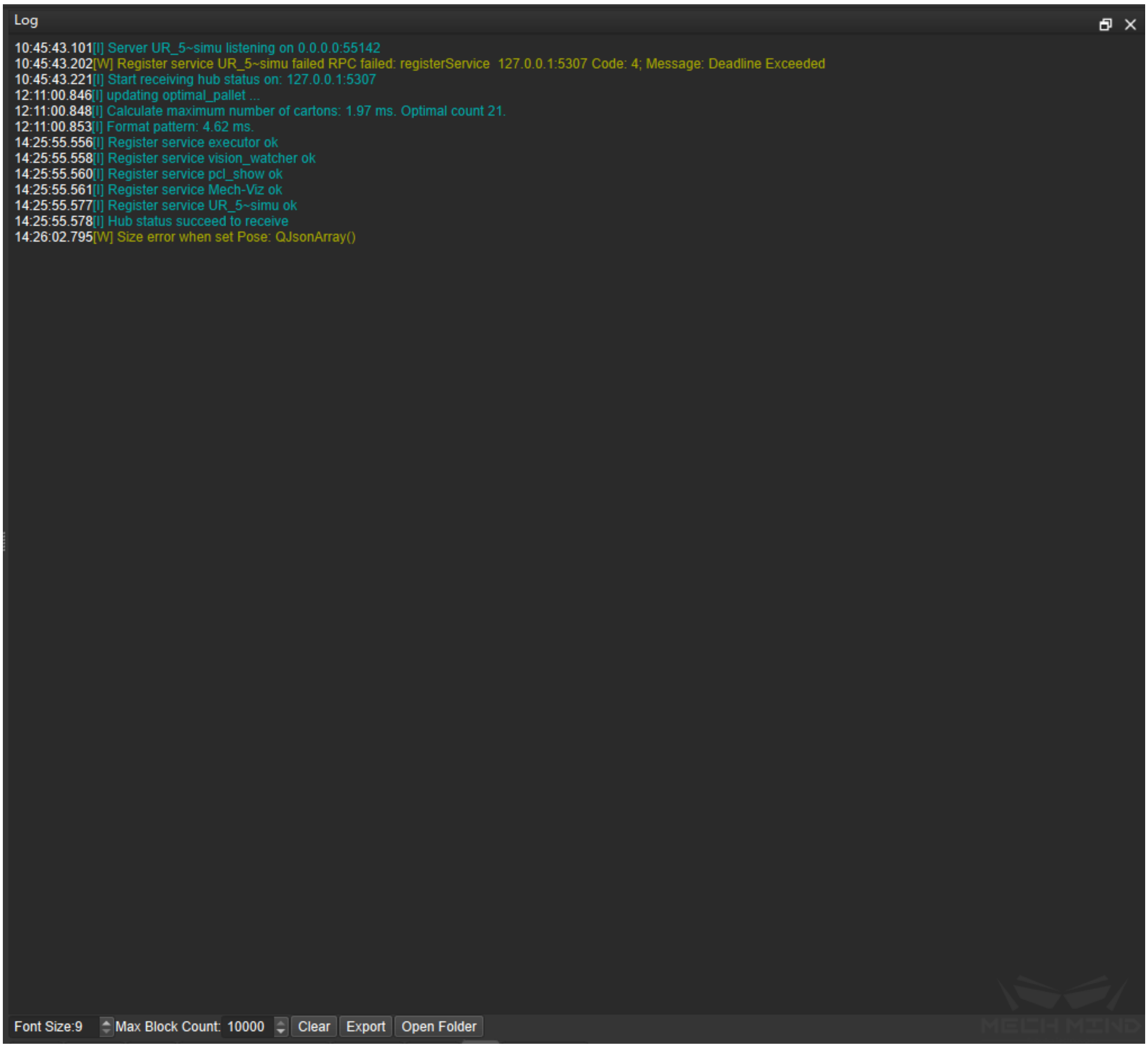


Figure 7 Input and output port of module

3. Select a module in programming area, the brief introduction and the parameters of this module will shown in the parameter panel, you can adjust this module’ s parameters in this area. Meanwhile, you can also set the global parameters such as: collision checking and runtime config in this area. More details about glocal parameters setting, please see *Tabs*

Log Tab

As shown in *Figure 8*the log tab is used to display log information during software operation, making it easy to debug and quickly locate problems.



```
10:45:43.101[I] Server UR_5~simu listening on 0.0.0.0:55142
10:45:43.202[W] Register service UR_5~simu failed RPC failed: registerService  127.0.0.1:5307 Code: 4; Message: Deadline Exceeded
10:45:43.221[I] Start receiving hub status on: 127.0.0.1:5307
12:11:00.846[I] updating optimal_pallet ...
12:11:00.848[I] Calculate maximum number of cartons: 1.97 ms. Optimal count 21.
12:11:00.853[I] Format pattern: 4.62 ms.
14:25:55.556[I] Register service executor ok
14:25:55.558[I] Register service vision_watcher ok
14:25:55.560[I] Register service pcl_show ok
14:25:55.561[I] Register service Mech-Viz ok
14:25:55.577[I] Register service UR_5~simu ok
14:25:55.578[I] Hub status succeed to receive
14:26:02.795[W] Size error when set Pose: QJsonArray()
```

Font Size:9

Max Block Count: 10000

Figure 8 Logs

2.1.4 Control Toolbar

The control toolbar is used to realize the simulation of the robot’s program and control the communication as well as connection with the real robot. The main functions are shown in the table

Icon	Option	Description
▶ Simulate	Simulate	Make the virtual robots perform actions according to designed tasks
Run	Run	Make the real robots perform actions according to designed tasks
■ Stop	Stop	Terminate an existing task
□ Keep Last Exec State	Keep Last Exex State	It is used for debugging the palletizing scene, and continues to stack from the position that has been palletized.
Reconnect	Reconnect	Establish a connection with Mech-Center to achieve data communication with the robot
Sync Robot	Sync Robot	Synchronize a virtual robot into a pose of a real robot
None	Robot Speed	Adjust the motion speed of robot with the slider or input box

2.1.5 Menu Bar

The contents of the menu bar are shown in the table

Tabs	**Description**
File	Contains open projects, save projects, save projects as…, recent projects, open projects in explorer, etc.
Edit	Operations about project editing
Setting	Related settings of the Mech-Viz system
View	Includes various control options of display
Tools	Contains various options of setting parameters
Help	Contains option for switching language, version information and log level.

Note: For more detail about the using of menu bar, please see *Menu Bar*

2.1.6 Keyboard and Mouse Operations

Mouse

Single-Click

Clicking the left mouse button is used to select or activate the button on the interface. When a module is clicked, the module frame will be highlighted and the robot’s pose or palletizing position and palletizing type of the current module will be displayed in the 3D simulation area.

Double-Click

- Double-clicking the model in the 3D simulation area or double-clicking the model name in the scene tab will bring up the edit box to adjust the model pose;
- For multi-level mission, double-clicking the mission module can expand the lower-level interface.

Right-Click

- Right-clicking in the programming area of the mission tab can bring up the interface as shown in *Figure 9*. It is mainly used for quick operations such as switching up and down levels in multi-level missions, resetting external signal states, and neatly arranging graphical modules;

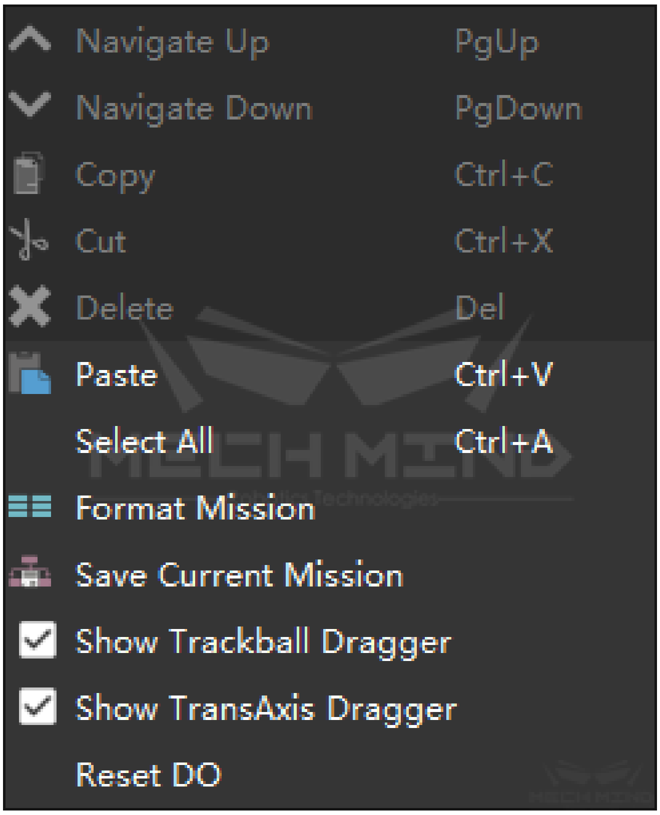


Figure 9 Right-click shortcut menu

- When you right-click a module, you can also perform basic operations such as copying and pasting the module.

Mouse Wheel Click

Pressing and holding the mouse wheel in the 3D simulation area can translate the viewing angle of the display area.

Scroll

- Rolling the mouse wheel in the 3D simulation area can zoom in and out of the entire display area.
- Holding down “Ctrl” and rolling the mouse wheel in the programming area can zoom in and out the display of program.
- In the parameter panel, the content of any edit box can be quickly resized by the mouse wheel.

Drag

- In any position of the 3D simulation area, holding down the left mouse button and dragging it can adjust the viewing angle of the display area; pressing Ctrl + left mouse button on the dragger to drag can adjust the pose of the robot;
- In the mission editing area, holding down the left mouse button can drag the module to any position in the programming area and drag the connection. You can also drag the slider to change the size of each joint angle of the robot or Euler angle in TCP coordinates.

Hot Keys

As shown in the table, the keyboard hot keys are the same as the system.

Hot Keys	Function
F5	Simulate
Ctrl+R	Run
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+Z	Rollback operation
Ctrl+Y	Resume operation
Ctrl+O	Open project
Ctrl+S	Save project
Delete	Delete
PgUp	To high-level of the multi-level mission
PgDn	To low-level of the multi-level mission

2.2 Menu Bar

2.2.1 File (F)

File (F) is for project management.

Options	Description	Shortcuts
Open Project	Open a project folder by a specified path.	Ctrl + O
Save Project	Save changes to the current project.	Ctrl + S
Save Project to JSON	Save the project and save the .viz file in the project folder as a .json file.	Ctrl + Shift + S
Save Project As⋯	Save the project to a specified path.	None
Recent Projects	Expand to display names of projects recently opened, click on the project name to open it directly.	None
Open Project In Explorer	Open the folder where the current project is located.	None
Open Executable File In Explorer	Open the folder where the Mech-Viz software currently running is located.	None
Backup Project	Backup the current project to a specified path.	None
Close Project	Close the current project in the Mech-Viz window.	None
Exit	Close and exit Mech-Viz.	Ctrl + Q

Note: By default, the project currently running is saved as a .viz file in Mech-Viz 1.4.0, and as a .json file in Mech-Viz of versions earlier than 1.4.0.

- The .viz format adopts binary encoding, so a .viz file can not be opened or edited directly by Notepad and other similar editor tools, which helps improve security and prevent error.
- The installation of Mech-Viz automatically registers means of opening and icon path of .viz files. In a project folder, the .viz file icon is changed to the Mech-Viz icon.
- The Mech-Viz software can be started and the project can be loaded automatically by double-clicking the .viz file.
- If using Mech-Viz of version 1.4.0 or above to open a project saved with Mech-Viz of versions below v1.4.0, the original .json project file will be automatically saved as and replaced by a .viz file after saving, and will be backed up as a .bak file.
- Click *File* → *Save Project To JSON* to save the project in .json as in earlier versions of Mech-Viz is necessary.

2.2.2 View (V)

View (V) is for changing the related settings of the interface display. By selecting a drop-down menu option, the corresponding tab will be displayed at the bottom right of the Mech-Viz interface, or the content of the interface will change accordingly.

Options	Description	Shortcuts
Full Screen	The current window will be displayed fullscreen when selected.	Ctrl + Shift + F11
Scene	Selected by default. The **Scene** tab will be displayed when selected.	None
Mission	Selected by default. The **Mission** tab will be displayed when selected.	None
Robot	Selected by default. The **Robot** tab will be displayed when selected.	None
End Effector & Objects	Selected by default. The **End Effector & Objects** tab will be displayed when selected.	None
Collisions	Selected by default. The **Collisions** tab will be displayed when selected.	None
Others	Selected by default. The **Others** tab will be displayed when selected.	None
Log	Selected by default. The **Log** tab will be displayed when selected.	None
Function Description	The function description of the selected skill will be displayed in the **Mission** window when selected.	None
Go Back	Go back to the last used page.	Alt + Left
Go Forward	Re-enter the next page used. It is the reverse operation of Go Back.	Alt + Right

2.2.3 Display (D)

Display (D) is for adjusting the details displayed in the 3D simulation on the left part of the Mech-Viz interface.

Options	Description
Show Received Vision Poses	Selected by default. The object poses received from Mech-Vision will be displayed when selected.
Show Point Cloud	Selected by default. The incoming point cloud image from the point cloud service will be displayed when selected.
Show Picked Object	Selected by default. The object model with measured dimensions will be displayed when selected.
Show Collision While Planning	Selected by default. The collision position(s) detected during planning in the 3D simulation will be displayed when selected.
Show Octree	The object point cloud will be displayed in the form of an octree when selected.
Show Received Carton Models	Selected by default. The received carton model(s) will be displayed in the 3D simulation when selected.
Show Arrow Pointing to Center	An arror point to object center(s) will be dispalyed when selected.
Show Object Pose	The object pose(s) will be displayed in the 3D simulation when selected.
Pose Status Colors	A window showing the colors representing different pose states will be displayed when selected.
Show Executor Status	The end effector’s state will be displayed at the task module icon in the mission window when selected.

2.2.4 Tools (T)

Tools (T) is for setting some assistive features of the software, mostly for debugging.

Options	Description	Shortcuts
Use Saved Vision Records (Only when Vision Service Not Registered)	The saved vision records are used for running the project when selected.	None
Set Vision Records	Save and retrieve visual records to reproduce issues during debugging.	None
Find Vision Results	Highlight the visual result corresponding to the searched serial number in the 3D simulation.	None
Not Print Message Sent to Robot	Disable the printing of the information sent by Mech-Viz to the robot in the log.	None
Write Debug File (.dmp)	Generate .dmp files for developers to troubleshoot problems.	None
Undo	Undo the last operation.	Ctrl + Z or Alt + T + U
Redo	Redo the last operation.	Ctrl + Y or Alt + T + R

2.2.5 Settings (S)

Settings (S) is for changing the common settings of the software.

Options	Description	Shortcuts
Set Mech-Center Address	Set the IP address of Mech-Center software.	None
Lock/Unlock Project	Lock the current project when selected.	None
List Frequently Used Properties	Only frequently-used parameters are displayed in the Mission window.	Ctrl + Shift + P
List Frequently Used Skills	Only frequently-used skills are displayed in the Mission window.	Ctrl + Shift + T
Options	Used to change the basic software settings (language, unit, etc.).	None
Log Level	Set the log level, the console will print logs of the current level and above.	None

2.2.6 Help (H)

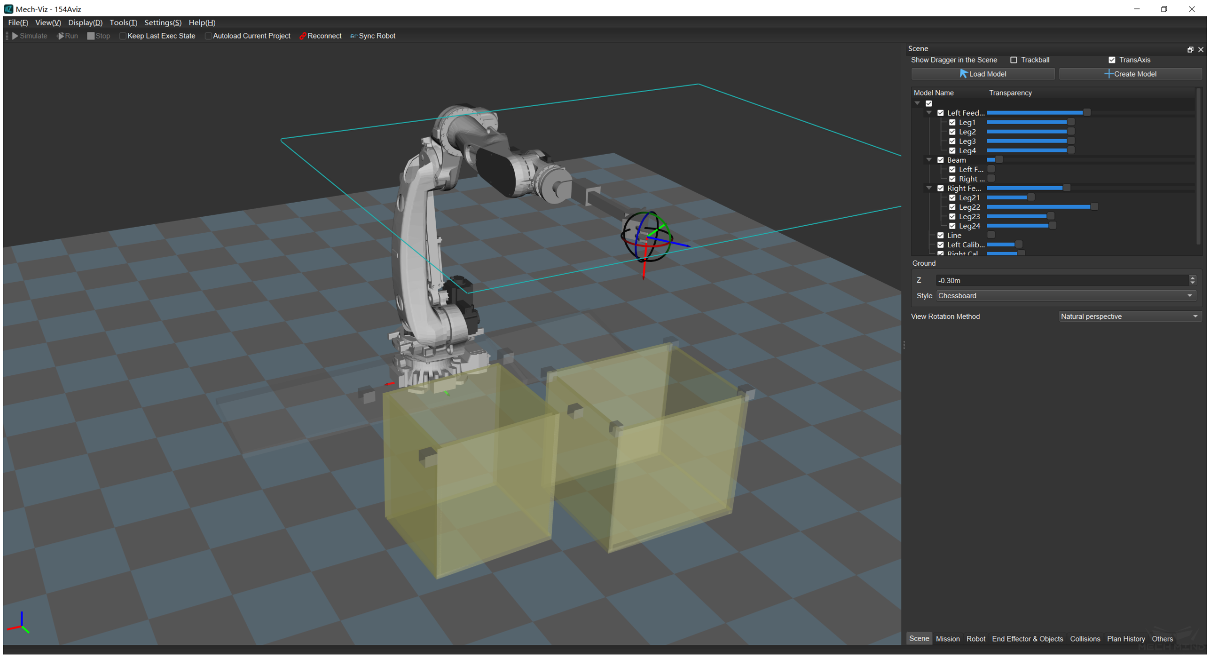
Help (H) is for viewing the current software version update.

Options	Description
About	Used to check the current version number.
Change Log	Open the change log page.

2.3 Tabs

2.3.1 Scene

Scene display settings can be adjusted in the interface as shown in *Figure 1*.



Show Dragger in the Scene □ Trackball ☑ TransAxis

Z -0.30m

Style Chessboard

View Rotation Method Natural perspective

Figure 1. Scene Tab Interface

Display the dragger (rotation, translation) of the objects in the scene

After checking, the draggable trackball and translation axes will be displayed respectively by clicking on the object in the scene. Please press and hold Ctrl and drag to perform translation or rotation of the object, as shown in *Figure 2*.

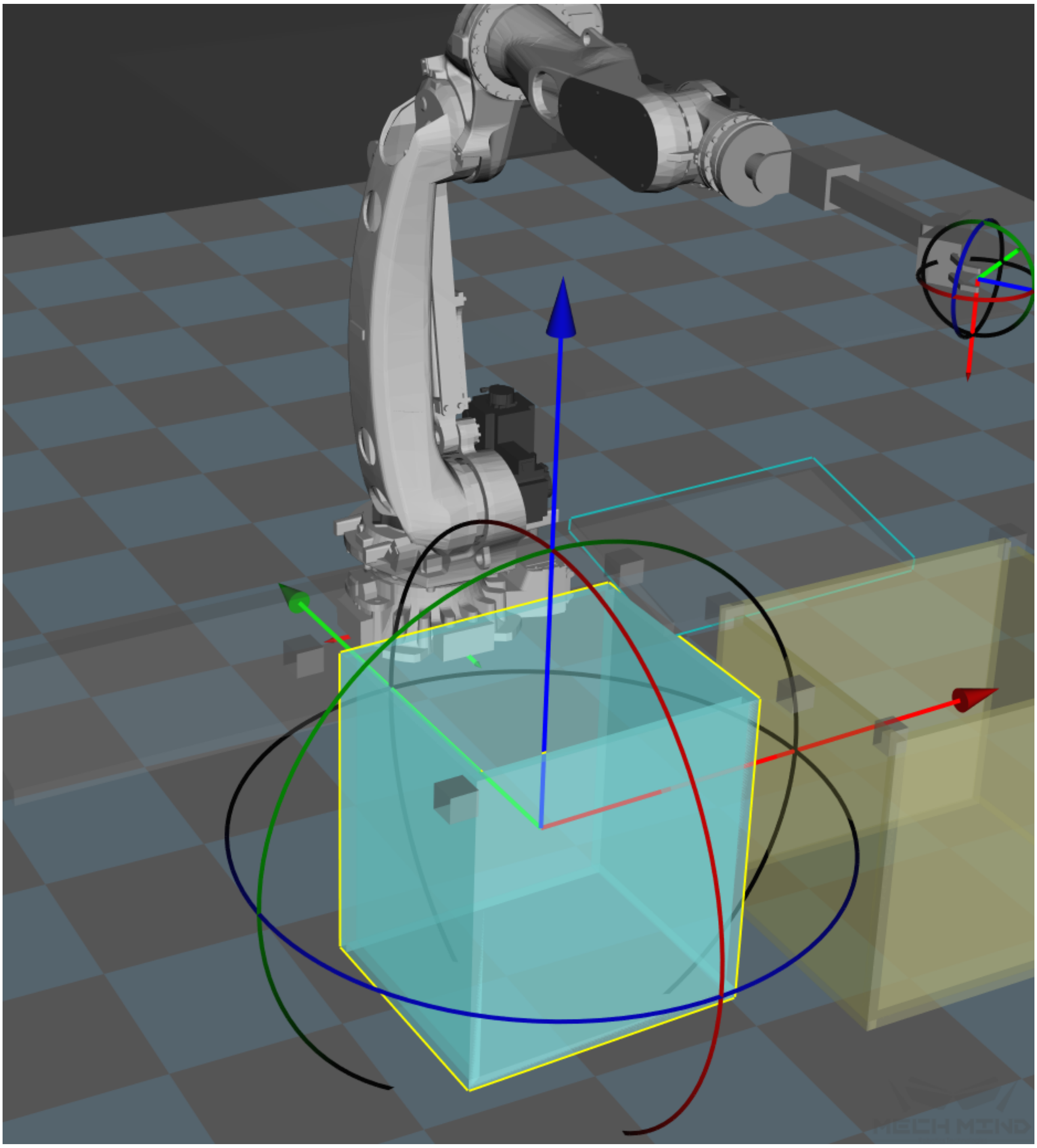
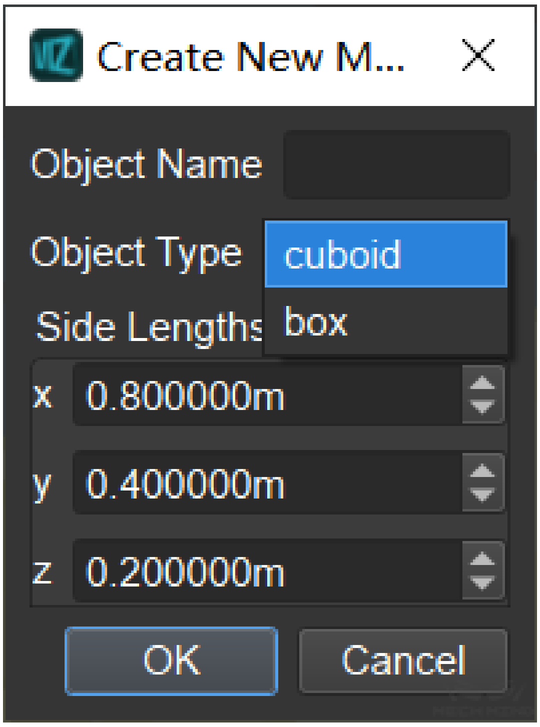


Figure 2. Draggers

Load Model, Create Model:

Load Model and Create Model are to load the built model or newly build a model. In Create Model, the type and dimensions of a new object can be set, as shown in *Figure 3*.



Object Name

Object Type: cuboid

Side Lengths

x 0.800000m

y 0.400000m

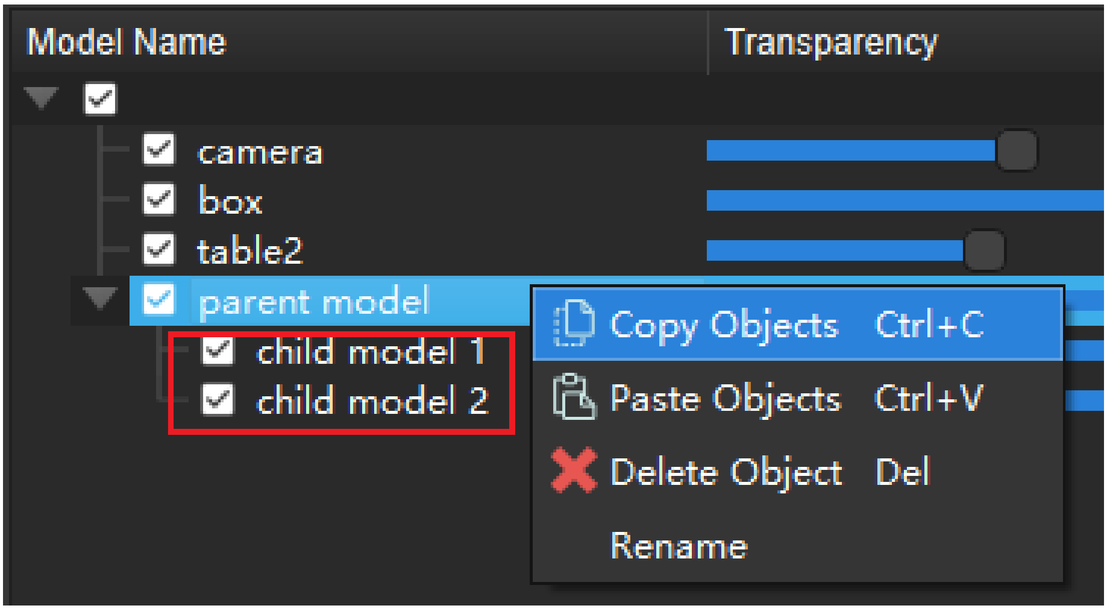
z 0.200000m

Figure 3. Create a New Model

Attention: After loading a new model, it may look larger than the scene. The reason is the unit has not been converted from meter to millimeter. To solve the issue, please double-click the model and scale the model by 0.001 in the model affine.

Copy and paste of scene model:

- Right-click the model name press shortcuts Ctrl+C and Ctrl+V to copy and paste.
- The child model is copied and pasted together with the parent model.
- After copying, click on the blank space to cancel the selection, and the model can be pasted in the root directory.
- The names of all model copies are concatenated with ‘-1’ ,’ -2’ ,’ -3’ ,’ -4’ , etc. on the back, and the number increases from the largest number in the exiting names of copies of the model.
- Press Ctrl for multiple selections, Ctrl+Z to undo, Ctrl+Y to redo.



Transparency setting:

Check the box on the left of the model name to show or hide the model, and drag the horizontal bar on the right to adjust the transparency of the models in the simulation on the left, as shown in *Figure 4*.

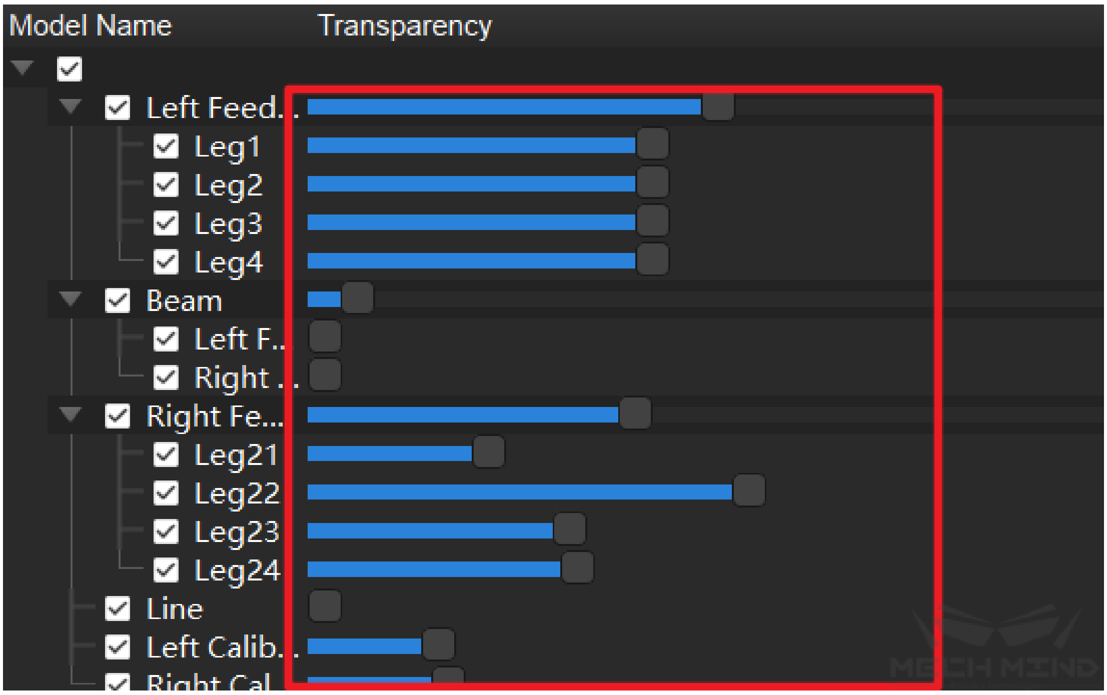
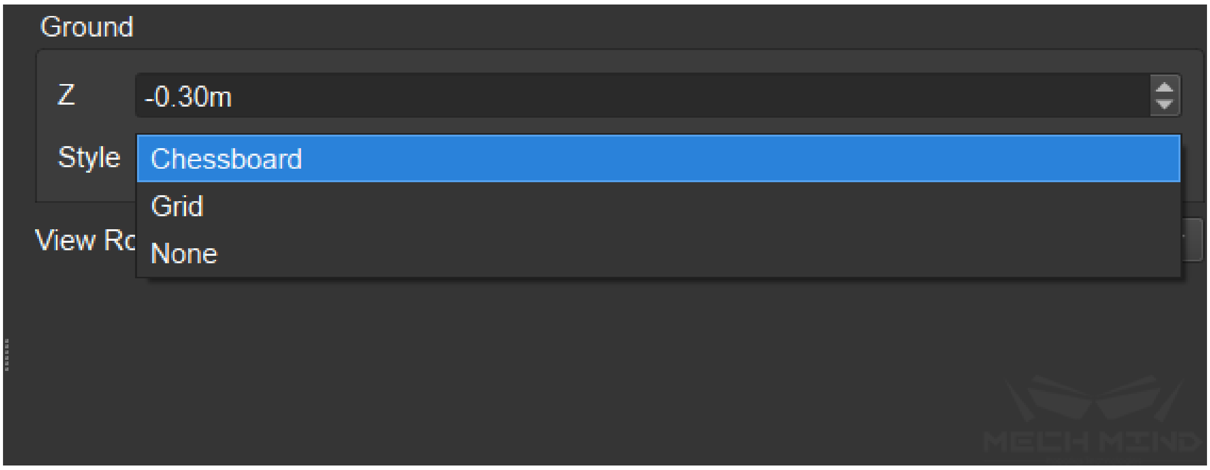


Figure 4. Transparency Settings

Ground settings:

The height and style of the ground in the scene can be set, as shown in *Figure 5*.



Ground

Z -0.30m

Style Chessboard

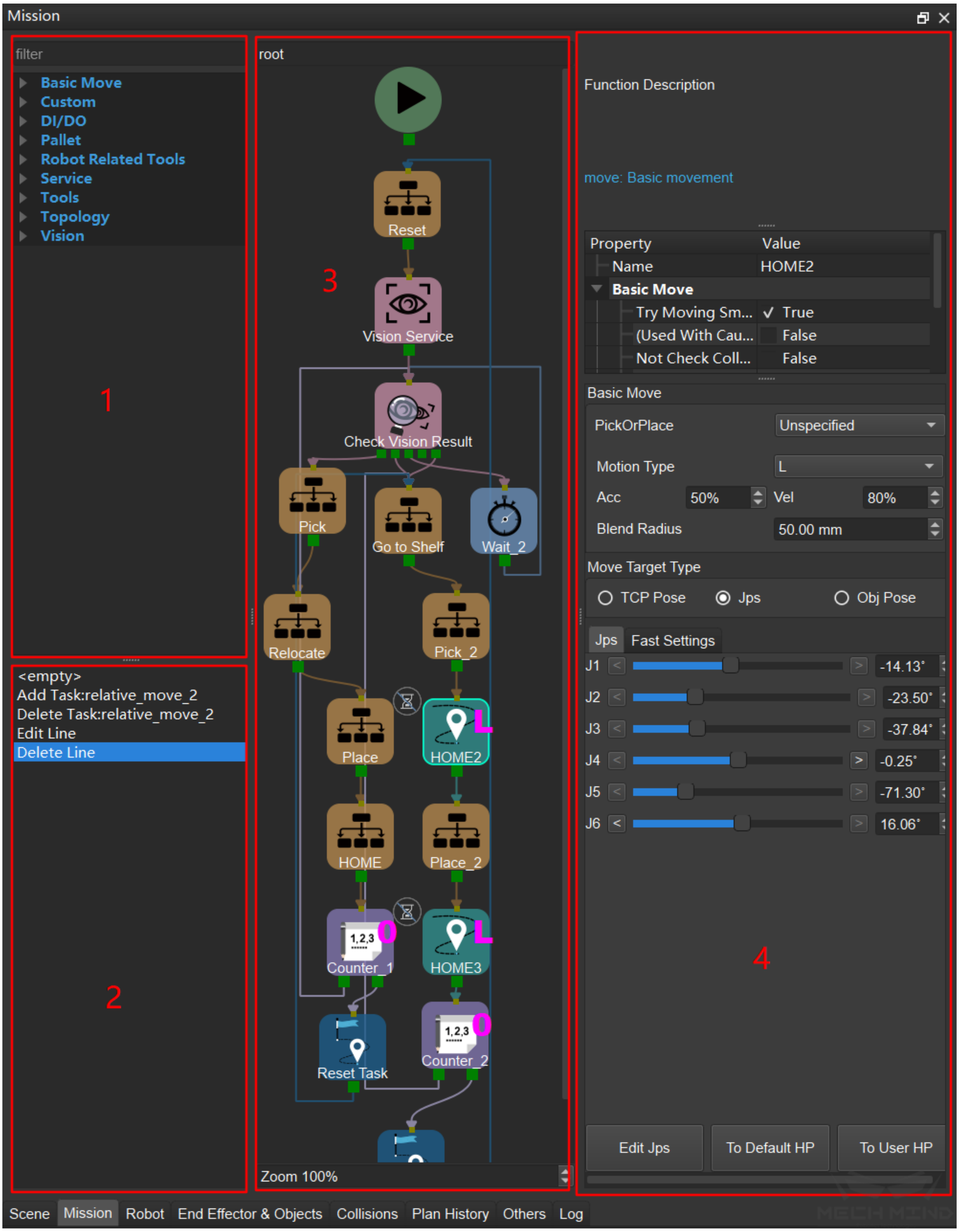
Figure 5. Ground Settings

View rotation method:

The view rotation method can be set as natural perspective or free perspective. Under both methods, the perspective can be adjusted by holding the left mouse button and dragging, and the line of sight is always towards the origin of the natural coordinate system, but with free perspective, the simulation can be rotated with the line of sight that passes the origin of the natural coordinate system as the axis.

2.3.2 Mission

The mission tab is the main place for robot programming. This page is mainly divided into four areas: module area, operation history area, programming area and parameter panel, as shown in the figure below.



move: Basic movement

Property	Value
Name	HOME2
Basic Move	
Try Moving Sm...	✓ True
(Used With Cau...	False
Not Check Coll...	False

Basic Move

PickOrPlace: Unspecified

Motion Type: L

Acc: 50% Vel: 80%

Blend Radius: 50.00 mm

Move Target Type

○ TCP Pose ● Jps ○ Obj Pose

Jps | Fast Settings

J1	-14.13°
J2	-23.50°
J3	-37.84°
J4	-0.25°
J5	-71.30°
J6	16.06°

1. **Module Area**

   There are many graphical programming modules provided by Mech-Viz in the module area.

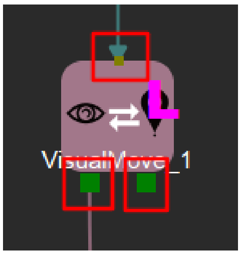
These modules are the basis for robot programming in Mech-Viz. Drag the module to the programming area, set the module properties, and then connect them in sequence according to the program operation logic to realize various robot programs. For detailed introduction and guidance for parameter settings of each module, please refer to *Skills*.

2. **Operation History Area**

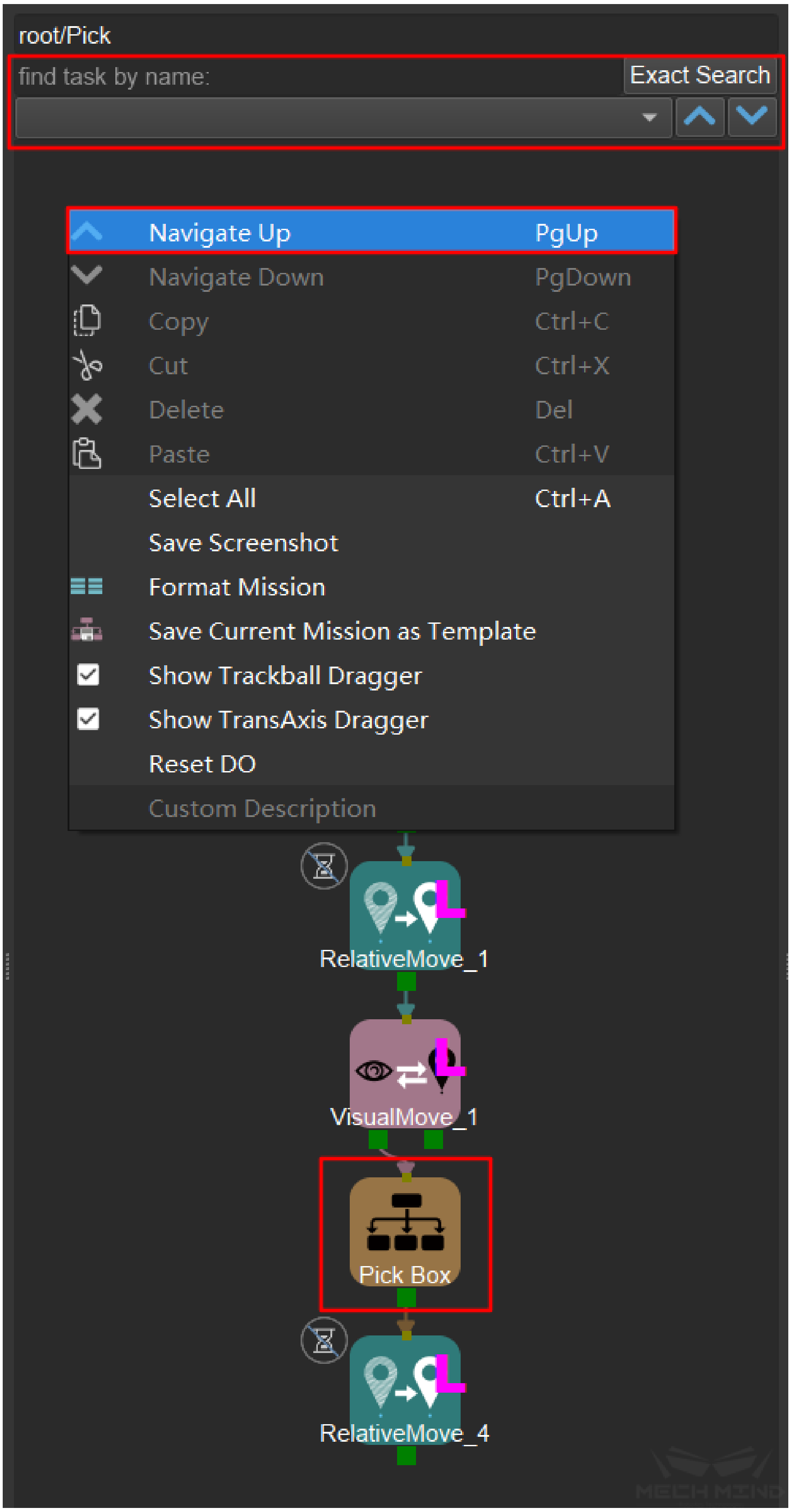
   Operations performed in the programming area are shown here, including adding modules, deleting modules, adding lines, deleting lines, etc. Click one of the records to roll back to the state of that record.

3. **Programming Area**

   The programming area is the area where the robot is programmed by combining various modules. The core is to select the appropriate modules and the correct connection between modules. Each module has an input port (small brown square) at the top, and one or more output ports (small green square) at the bottom. Lines can be drawn from the output port and connected to the input port.

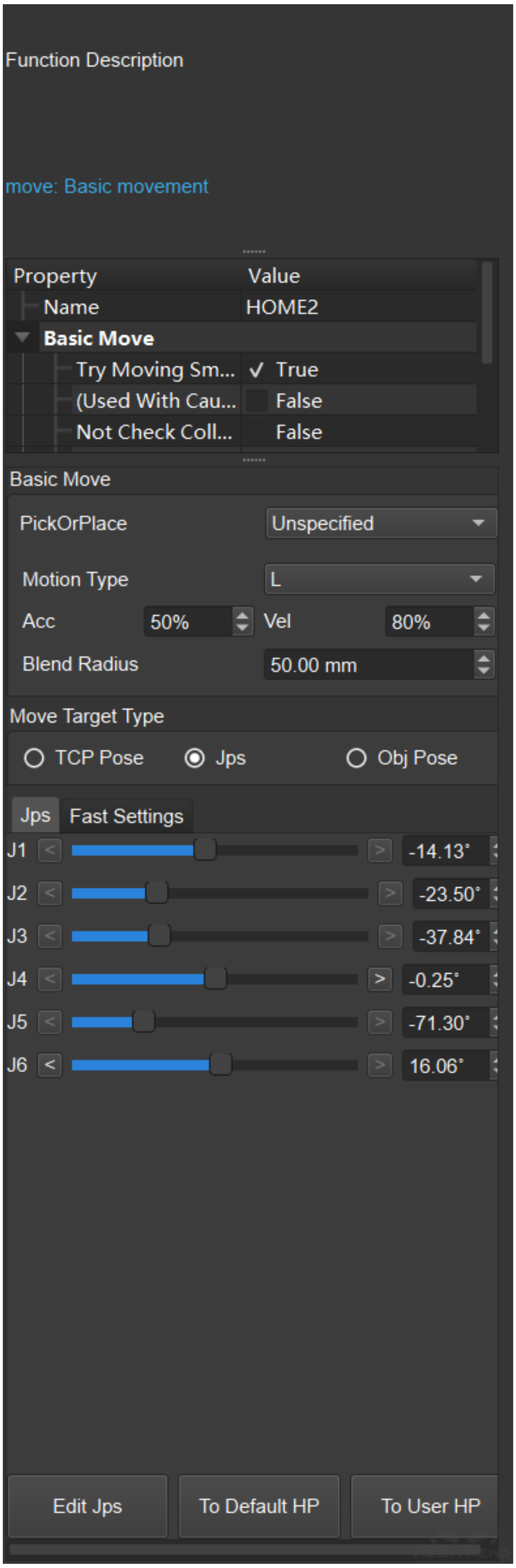


   Multiple modules can also be combined to form a task set, which is represented by a "mission" module on the main program, so that the program is more concise and clear. Double-click the task set to enter it for editing, right-click and select *Navigate Up* to return to the previous level. In addition, if you need to quickly find a specific module in the program, you can press the hotkey Ctrl+F. At this time, a search box will appear at the top of the programming area, and you can enter the module name to search.


root/Pick
find task by name:
Exact Search
Navigate Up PgUp
Navigate Down PgDown
Copy Ctrl+C
Cut Ctrl+X
Delete Del
Paste Ctrl+V
Select All Ctrl+A
Save Screenshot
Format Mission
Save Current Mission as Template
Show Trackball Dragger
Show TransAxis Dragger
Reset DO
Custom Description
RelativeMove_1
VisualMove_1
Pick Box
RelativeMove_4

4. **Parameter Panel**

   When no module is selected, there is nothing. When a module is selected, the function description and parameter settings of the selected module are shown under the speed settings.



Function Description

move: Basic movement

Property	Value
Name	HOME2
Basic Move	
Try Moving Sm...	✓ True
(Used With Cau...	False
Not Check Coll...	False

Basic Move

PickOrPlace: Unspecified

Motion Type: L

Acc: 50% Vel: 80%

Blend Radius: 50.00 mm

Move Target Type

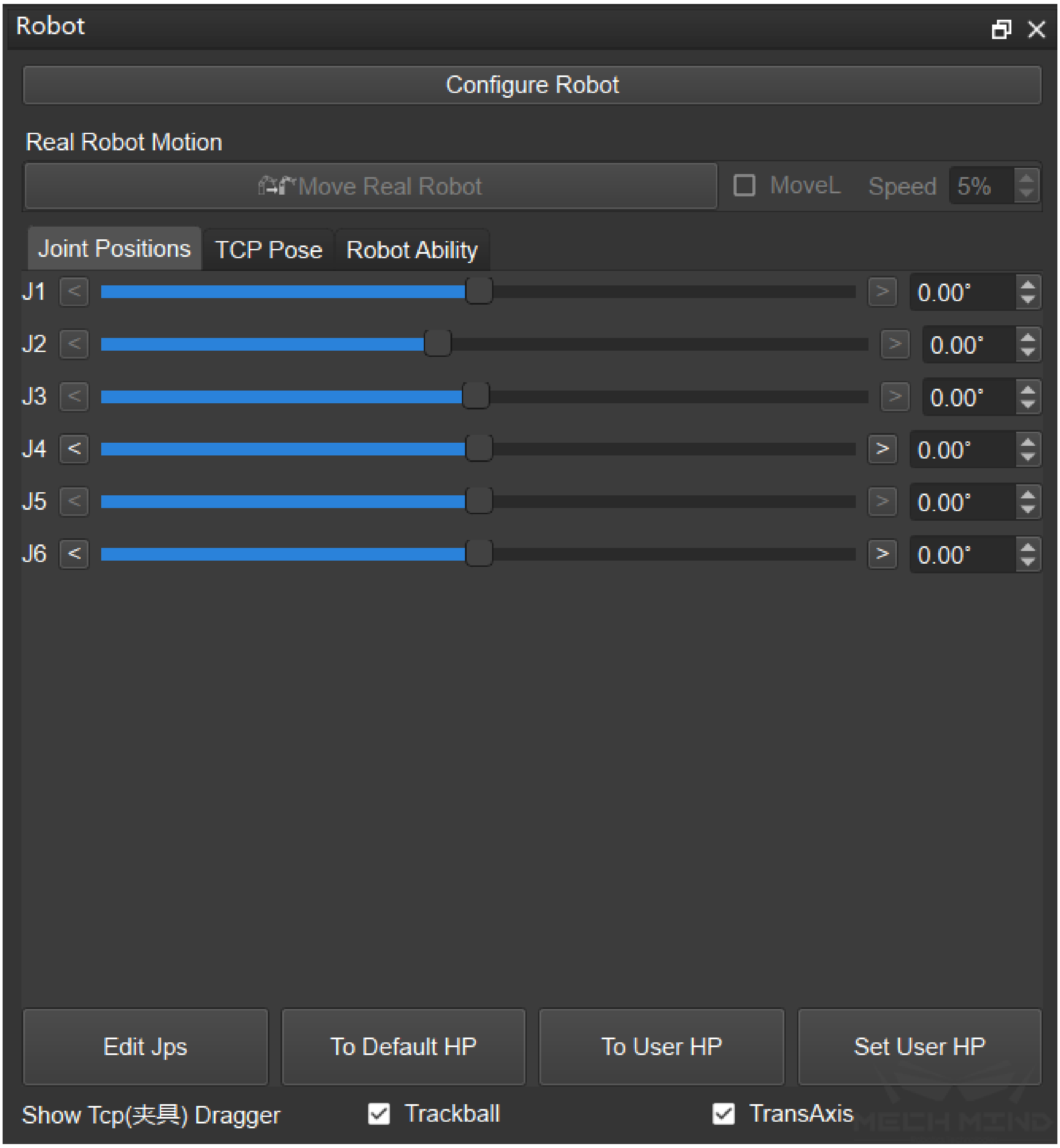
○ TCP Pose ● Jps ○ Obj Pose

Jps | Fast Settings

J1	-14.13°
J2	-23.50°
J3	-37.84°
J4	-0.25°
J5	-71.30°
J6	16.06°

2.3.3 Robot

The main content of the robot tab page is shown in *figure 1*. The function and usage of each option will be introduced below.



☐ MoveL

Speed 5%

J1 0.00°

J2 0.00°

J3 0.00°

J4 0.00°

J5 0.00°

J6 0.00°

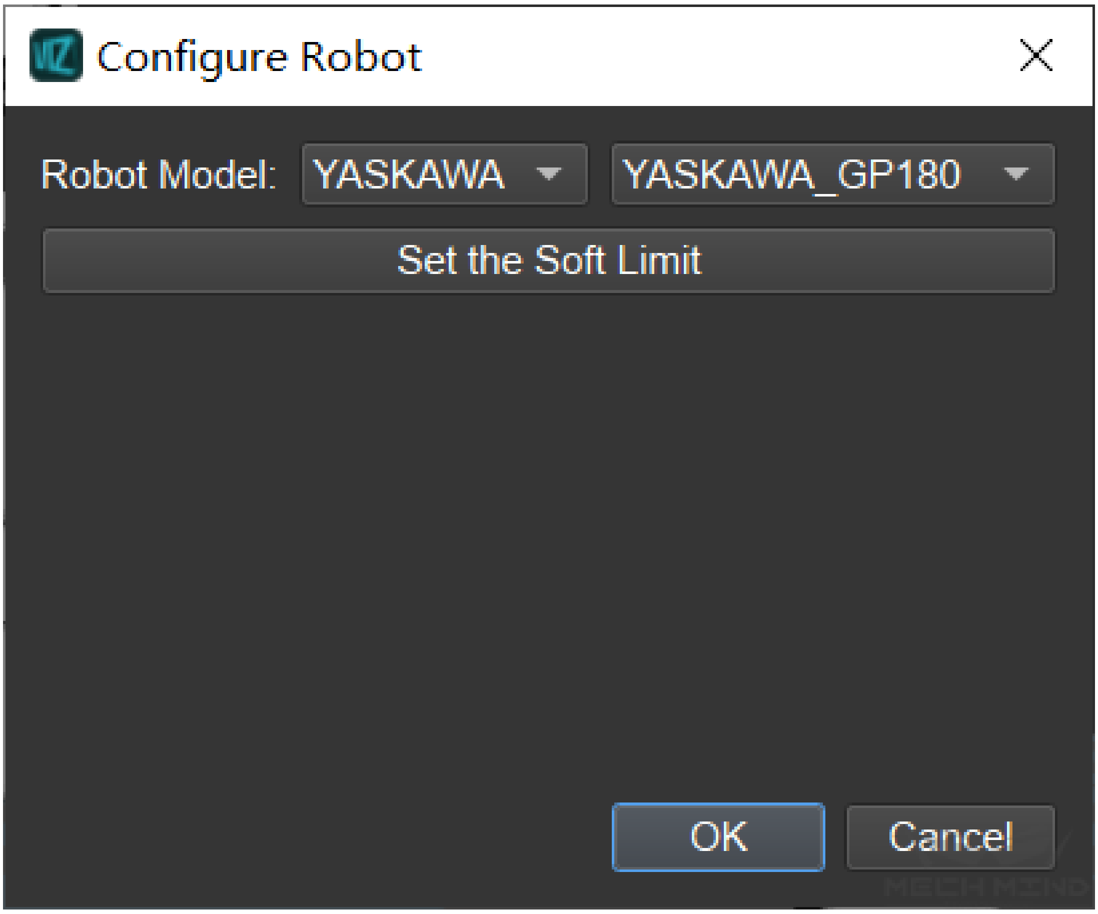
☑ Trackball

☑ TransAxis

figure 1 Robot tab page

Configure Robot

After clicking *Configure Robot*, you can import the robot model by selecting the vendor and model of the robot, as shown in *figure 2*.

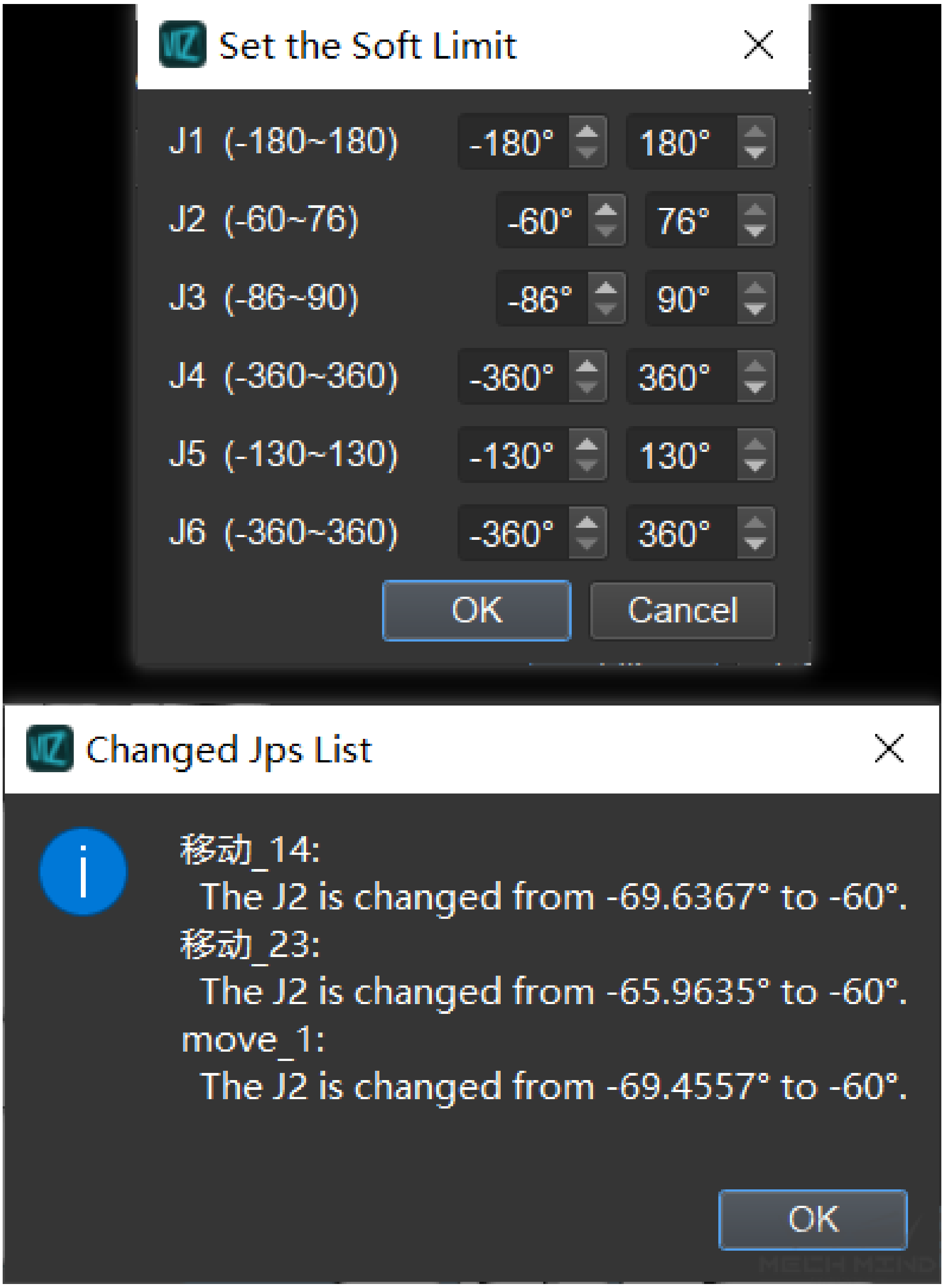


Configure Robot

Robot Model: YASKAWA YASKAWA_GP180

figure 2 Configure robot

Through *Set Soft Limit*, you can set the soft limit for each axis of the robot. When the soft limit range is less than the existing joint angle setting, a pop-up window will prompt and show the affected Skills, ss shown in *figure 3*.



Set the Soft Limit

J1 (-180~180) -180° 180°

J2 (-60~76) -60° 76°

J3 (-86~90) -86° 90°

J4 (-360~360) -360° 360°

J5 (-130~130) -130° 130°

J6 (-360~360) -360° 360°

Changed Jps List

移动_14:
The J2 is changed from -69.6367° to -60°.
移动_23:
The J2 is changed from -65.9635° to -60°.
move_1:
The J2 is changed from -69.4557° to -60°.

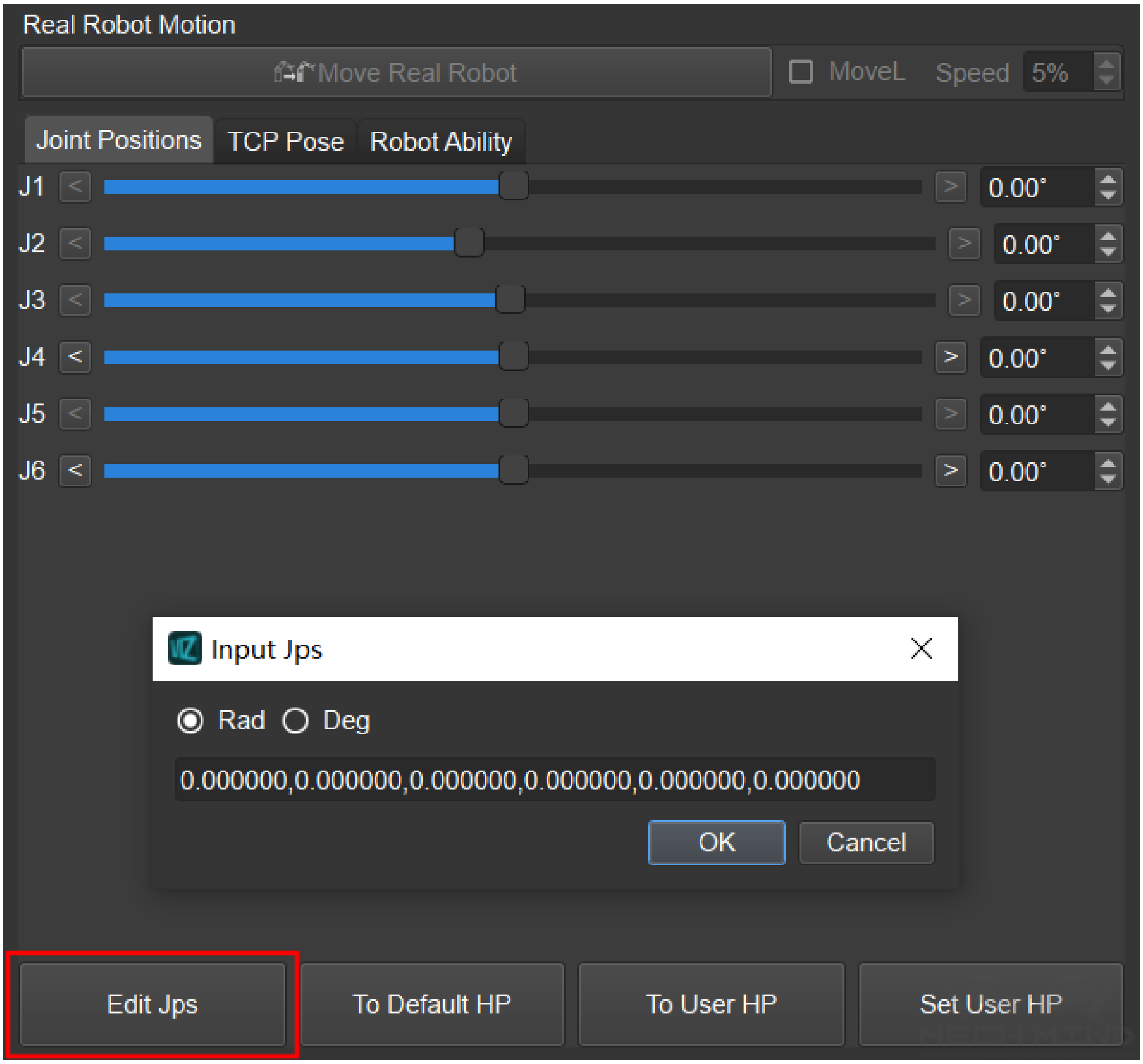
figure 3 Set soft limit

Real Robot Motion

Click *Move Real Robot*, the real robot will move to the current pose of the simulated robot. On the right side of *Mobile Real Robot*, you can also check the *MoveL* option and set its speed if needed.

Joint Positions

Under the joint positions option, the current joint positions data of the simulated robot is displayed. If you need to synchronize with the real robot, please click *Sync Robot* in the toolbar at the top of the software when the robot is connected. If you need to adjust the pose of the simulated robot, you can drag the slider of each axis or directly enter the value in the right input box, and the simulated robot will move in real time according to the adjustment. Click the *Edit Jps* button, you can enter the joint angle value of each axis in the pop-up window at one time, as shown in *figure 4*.



□ MoveL Speed 5%

J1 0.00°
J2 0.00°
J3 0.00°
J4 0.00°
J5 0.00°
J6 0.00°

Input Jps

● Rad ○ Deg

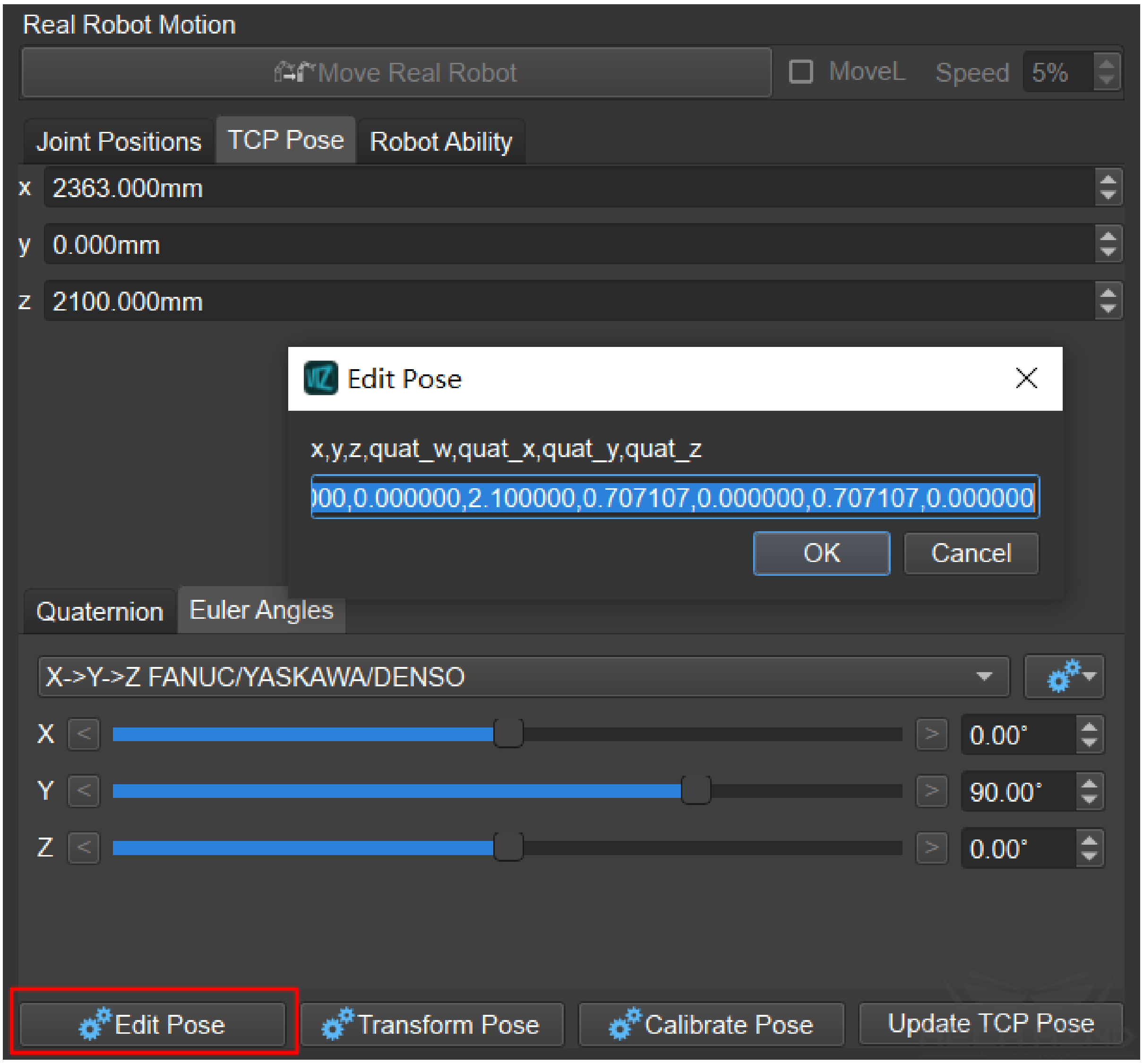
0.000000,0.000000,0.000000,0.000000,0.000000,0.000000

figure 4 Edit joint positions

Click the *To Default HP* button, and the simulated robot will return to the initial position set in the robot configuration file. Click the *To User HP* button, and the simulated robot will return to the initial position set by the user. This position is set by the *Set User HP* button on the right.

TCP Pose

Under the TCP pose option, the current TCP pose data of the simulated robot is displayed. The TCP pose can be adjusted through the following *Quaternion* and *Euler Angles* options, or you can directly input the TCP pose through *Edit Pose*, as shown in *figure 5*.



x 2363.000mm

y 0.000mm

z 2100.000mm

x,y,z,quat_w,quat_x,quat_y,quat_z

)00,0.000000,2.100000,0.707107,0.000000,0.707107,0.000000

X->Y->Z FANUC/YASKAWA/DENSO

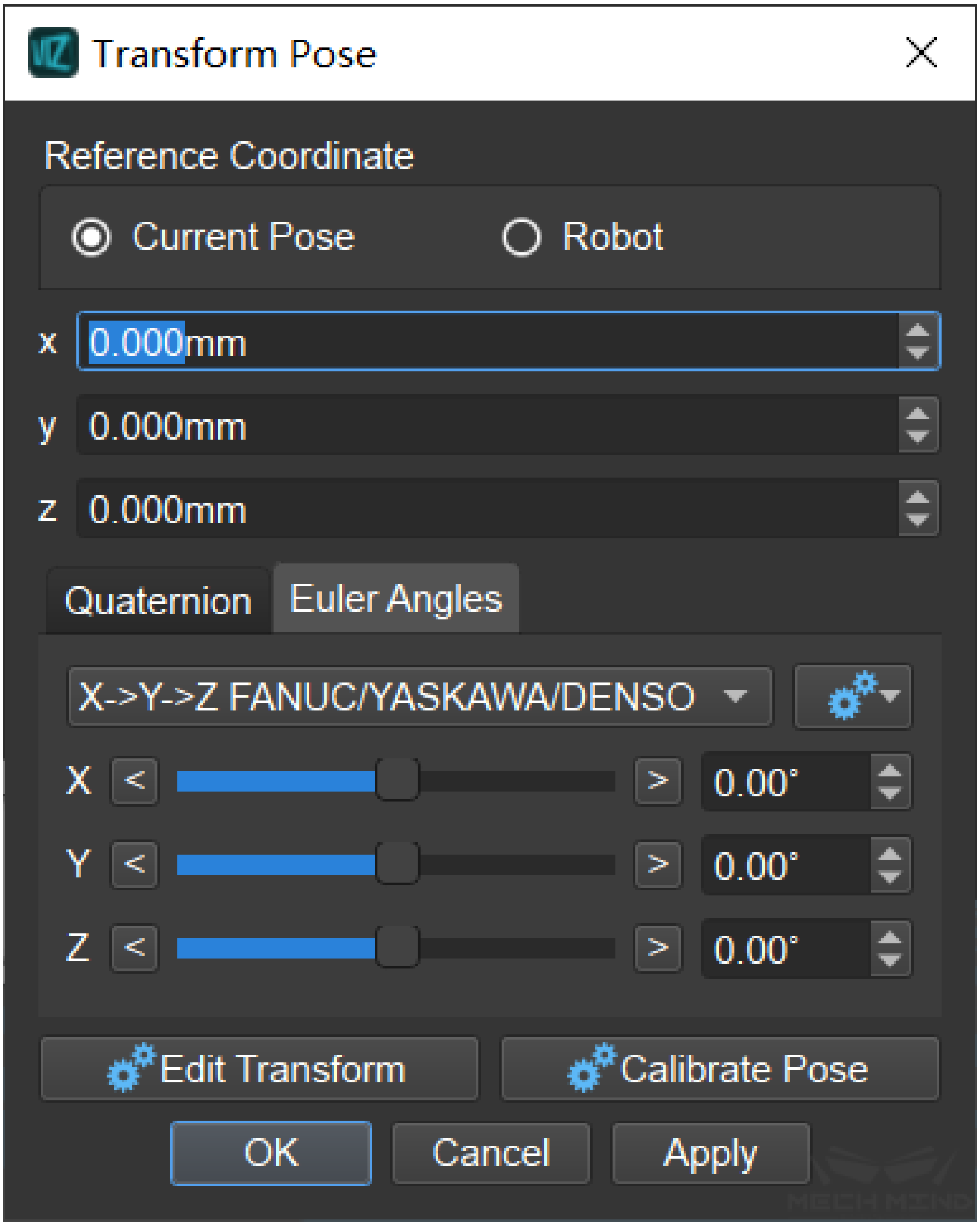
X 0.00°

Y 90.00°

Z 0.00°

figure 5 Edit pose

Click the *Transform Pose* button to transform the current pose to a new pose by defining a transformation. As shown in *figure 6*, you first need to select the reference coordinate system, and then define the transformation through XYZ coordinates, quaternion, Euler angles or *Edit Transform* button, the simulation robot will display the transformation result in real time. If you need to save the result, please click *Apply*; if you don’ t need to save the result, please click *Cancel* to exit, and the simulated robot will return to its original position.



Transform Pose

Reference Coordinate

● Current Pose ○ Robot

x 0.000mm

y 0.000mm

z 0.000mm

X->Y->Z FANUC/YASKAWA/DENSO

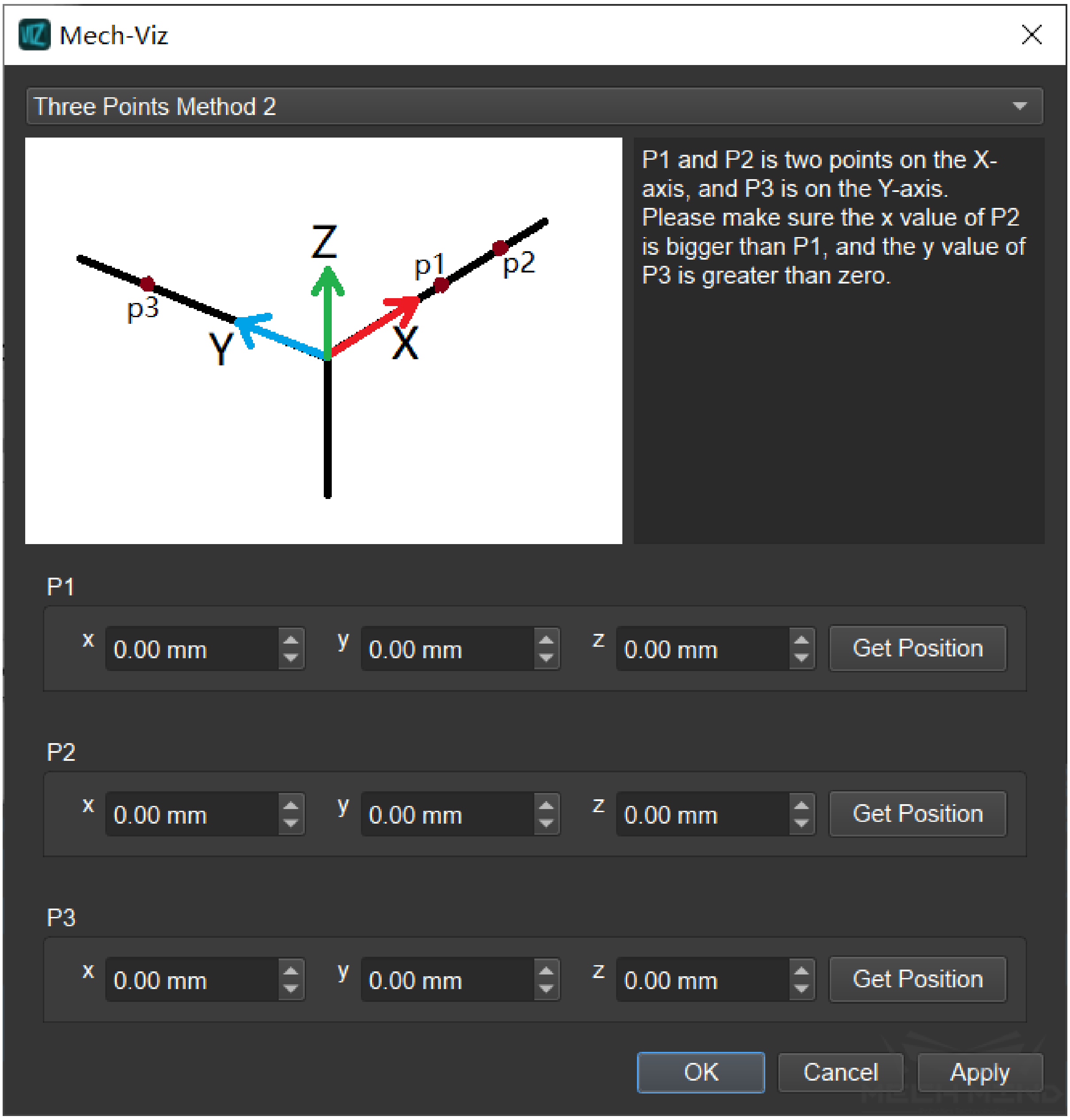
X 0.00°

Y 0.00°

Z 0.00°

figure 6 Transform pose

If you need to calibrate the TCP pose, please click the *Calibrate Pose* button, and then follow the pop-up prompts to calibrate, as shown in *figure 7*.



P1 and P2 is two points on the X-axis, and P3 is on the Y-axis. Please make sure the x value of P2 is bigger than P1, and the y value of P3 is greater than zero.

P1

x 0.00 mm y 0.00 mm z 0.00 mm Get Position

P2

x 0.00 mm y 0.00 mm z 0.00 mm Get Position

P3

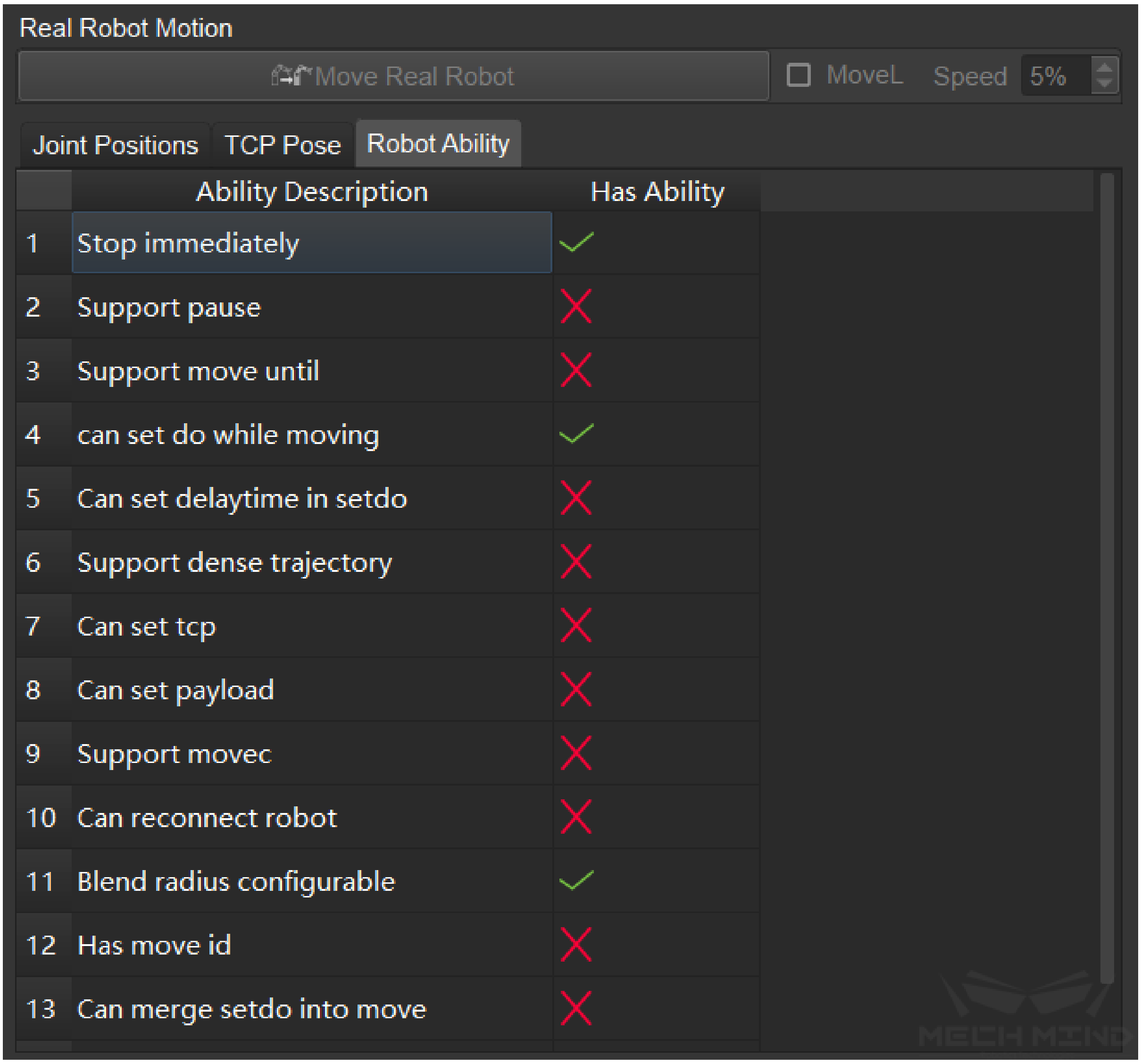
x 0.00 mm y 0.00 mm z 0.00 mm Get Position

figure 7 Calibrate pose

When the robot is connected, click *Update TCP Pose*, Mech-Viz will obtain the TCP pose from the real robot.

Robot Ability

This option shows whether the current robot has related functions, as shown in *figure 8*.



	Ability Description	Has Ability
1	Stop immediately	✓
2	Support pause	✗
3	Support move until	✗
4	can set do while moving	✓
5	Can set delaytime in setdo	✗
6	Support dense trajectory	✗
7	Can set tcp	✗
8	Can set payload	✗
9	Support movec	✗
10	Can reconnect robot	✗
11	Blend radius configurable	✓
12	Has move id	✗
13	Can merge setdo into move	✗

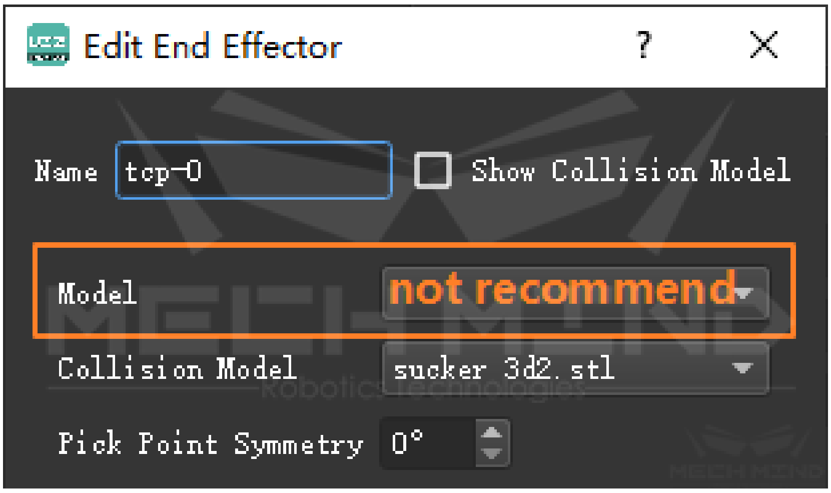
figure 8 Robot ability

Show TCP Dragger

This option refers to the dragger of the end effector on the simulated robot. If the Trackball and TransAxis options are checked, you will be able to intuitively adjust the TCP pose by directly dragging the end effector on the simulated robot.

2.3.4 End Effector & Objects

Add End Effector Models



Name tcp-0 □ Show Collision Model

Model not recommend

Collision Model sucker 3d2.stl

Pick Point Symmetry 0°

Projects that require point cloud collision detection must load end effector models. It is generally recommended to add only the collision models and not the display models.

Advantages:

1. The project folder will be smaller, because the storage size of the display model is about 3 times that of the collision model;
2. Editing will be easier. When the end effector is displaced, the collision model is only to be loaded and set once. If both the display model and the collision model are added, they shall be set separately;
3. The problem of inconsistency between the display model and the collision model is eliminated, and the situation that the collision detection is not as expected is avoided.

In addition, the collision model shall be as rough as possible, delete the details, and keep the contours required for collision detection only, otherwise it will affect the software calculation speed.

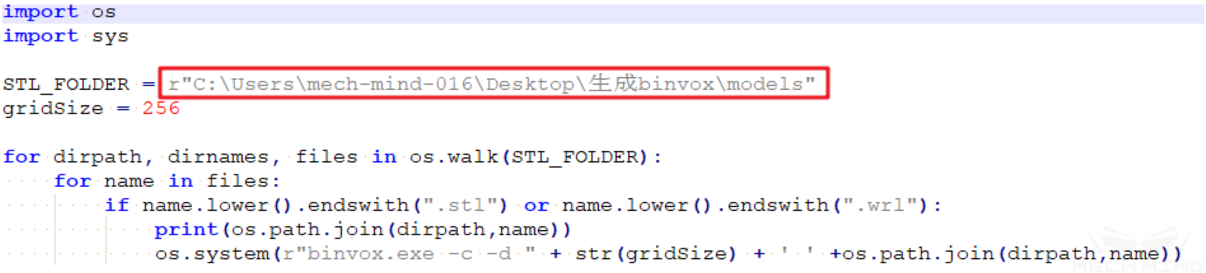
Collision Models of .binvox Format

For some scenario that needs high accuracy of point cloud collision checking, it is recommended to add model with .binvox format.

Save the model in the format of .stl or .wrl by using the CAD software like SolidWorks.

> The .binvox model generated from .stl model is only a frame.
>
> The .binvox model generated from .wrl model is a solid model.
>
> The degree of collision can be shown more specific if using solid model, such as touch or real collision, however, it also cost more time to calculate.

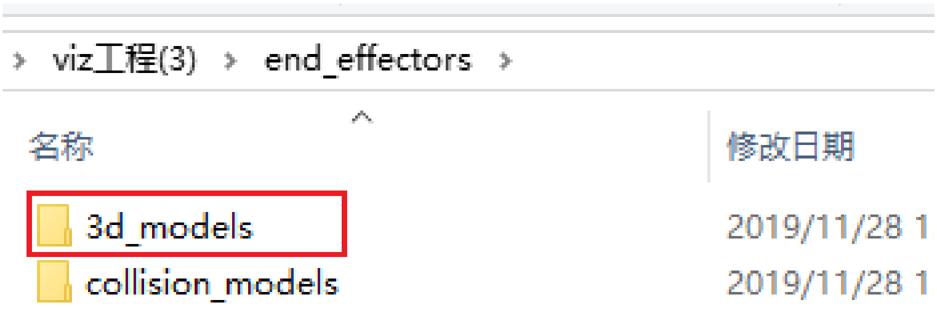
Download “generate_binvox.zip” and unzip it, open “generate_binvoxes.py” with notepad++, change the file path of the .stl or .wrl model as follows(do not delete “r”), double click to run it after saving.



```
import os
import sys

STL_FOLDER = r"C:\Users\mech-mind-016\Desktop\生成binvox\models"
gridSize = 256

for dirpath, dirnames, files in os.walk(STL_FOLDER):
    for name in files:
        if name.lower().endswith(".stl") or name.lower().endswith(".wrl"):
            print(os.path.join(dirpath,name))
            os.system(r"binvox.exe -c -d " + str(gridSize) + ' ' +os.path.join(dirpath,name))
```

After execution, the models in format .stl or .wrl are converted to the .binvox file.

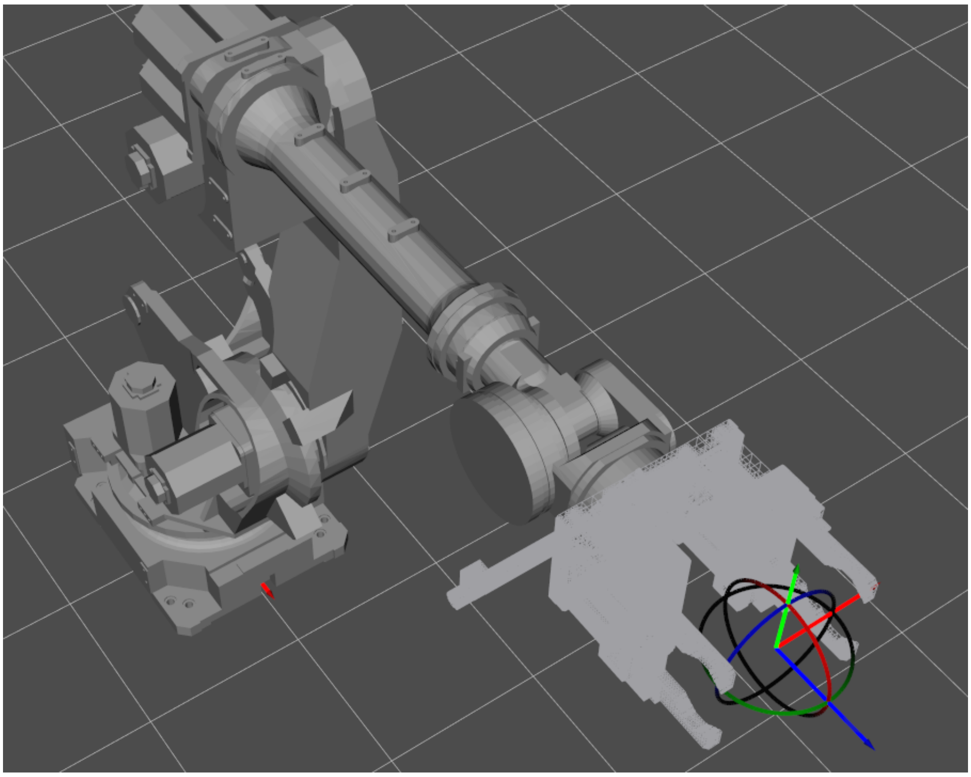
Attention: **Only the model generated by ascii format can be converted** , it is not support binary format. If the .stl model is in binary, you can add it from 3D Models tab in Mech-Viz, after saving, you can find the ascii-generated model in the path “end_effectors/3d_models”, as the figure below:



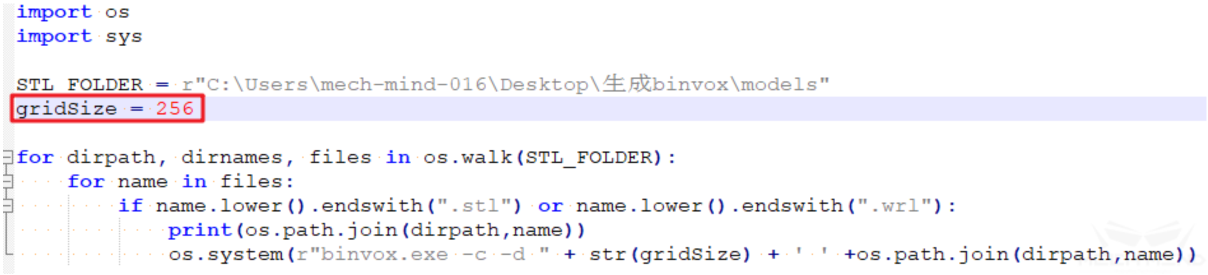
› viz工程(3) › end_effectors ›

名称	修改日期
3d_models	2019/11/28 1
collision_models	2019/11/28 1

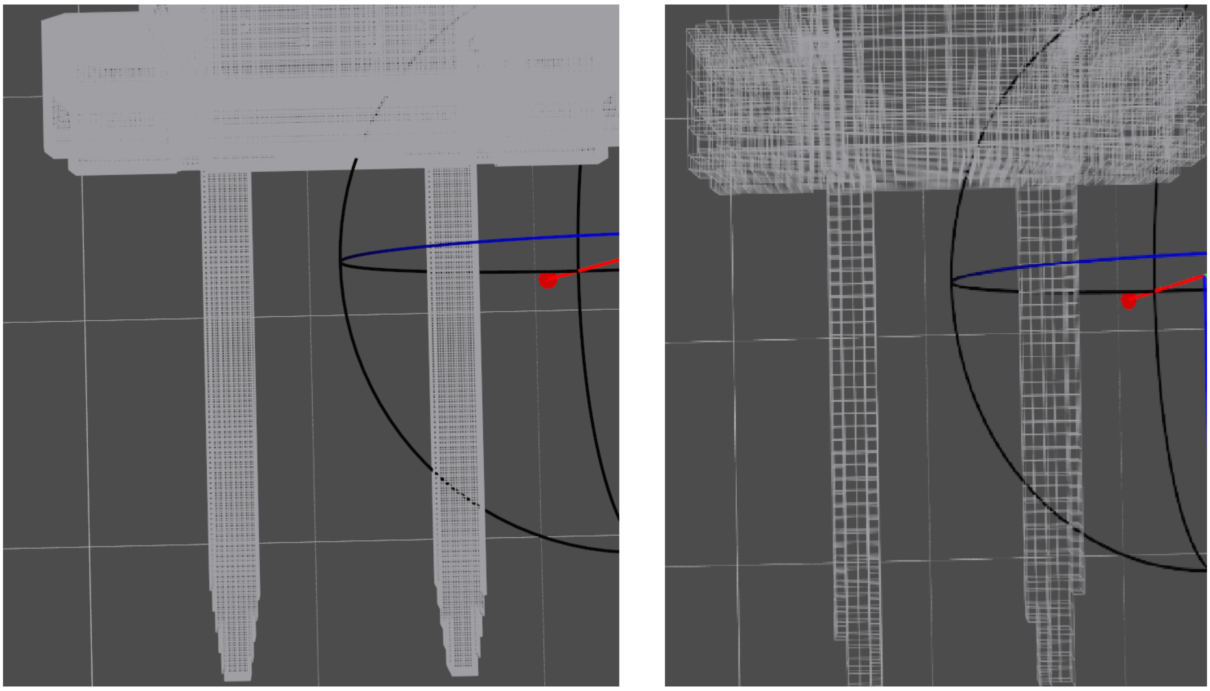
The loaded .binvox model in Mech-Viz:



The roughness of the .binvox model is affected by the value of gridSize:

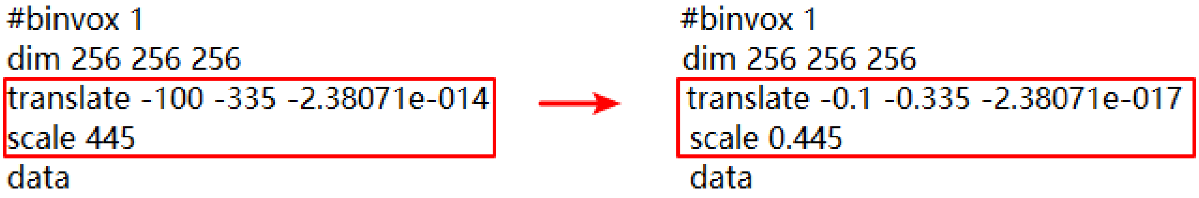


```
import os
import sys

STL_FOLDER = r"C:\Users\mech-mind-016\Desktop\生成binvox\models"
gridSize = 256

for dirpath, dirnames, files in os.walk(STL_FOLDER):
    for name in files:
        if name.lower().endswith(".stl") or name.lower().endswith(".wrl"):
            print(os.path.join(dirpath,name))
            os.system(r"binvox.exe -c -d " + str(gridSize) + ' ' +os.path.join(dirpath,name))
```



gridSize = 512 & gridSize = 100

If the loaded model in the scene is unexpected huge, that is because the unit is not proper. Open the .binvox model file by notepad++, reduce the values of scale and translate 1000 times, then load the model again.



```
#binvox 1
dim 256 256 256
translate -100 -335 -2.38071e-014
scale 445
data
```

```
#binvox 1
dim 256 256 256
translate -0.1 -0.335 -2.38071e-017
scale 0.445
data
```

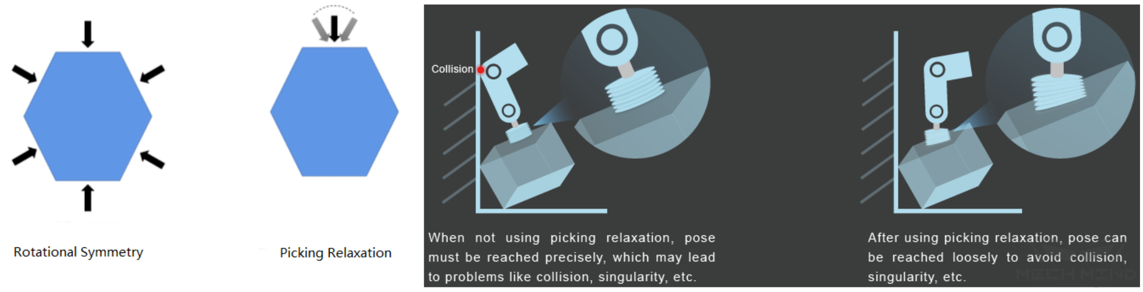
Configuration of Objects to Pick (for Move Planning)

Each object to be picked has an *Object Pose (Obj Pose)* , and the software will calculate the corresponding *TCP pose* and control the robot to pick it.

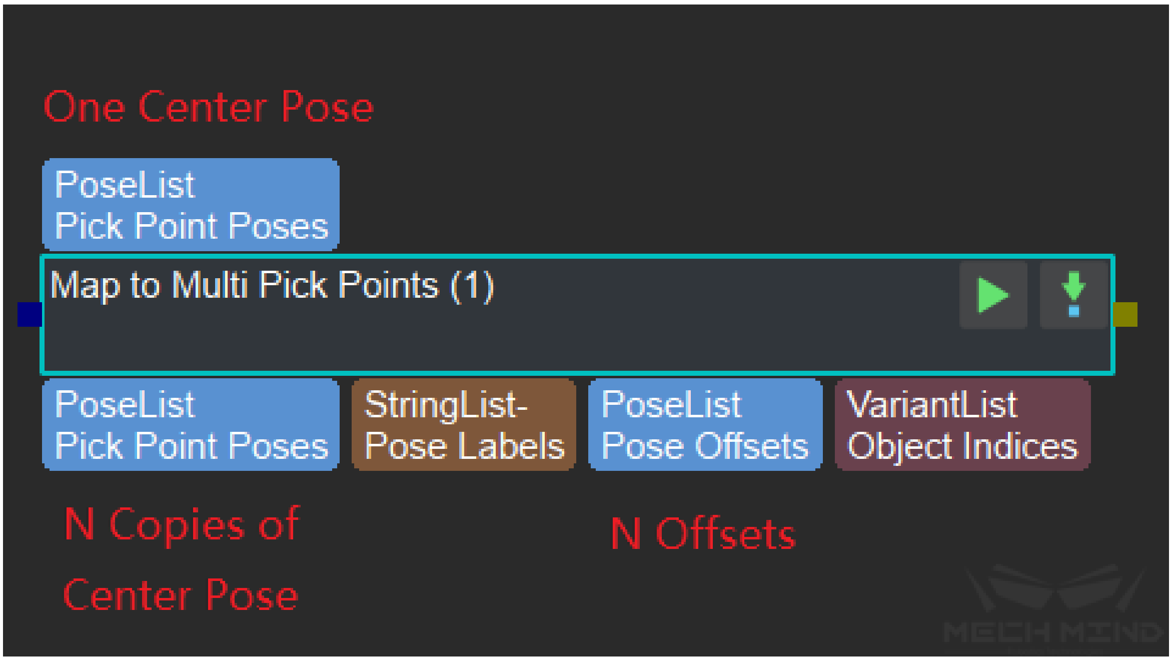
Symmetry and Picking Relaxation

Symmetry: The objects to be picked or the pick points are often symmetrical. Please visit *Symmetry* for more information.

Picking Relaxation: Due to the adaptability of the end effector (especially the sucker) or the object itself, a certain deviation between the end effector and the object can be allowed when picking. By setting the picking relaxation, the robot can actively use such "tolerable deviations" to avoid problems such as collisions and singularities. In principle, the picking relaxation is "tolerable deviation" based on the pick point. In order to obtain the optimal motion trajectory, the software will consider the symmetry of the pick point, the picking relaxation and the symmetry of the object together during the picking process; and the object pose will be selected according to the symmetry of the object when placing.


Rotational Symmetry
Picking Relaxation
Collision
When not using picking relaxation, pose must be reached precisely, which may lead to problems like collision, singularity, etc.
After using picking relaxation, pose can be reached loosely to avoid collision, singularity, etc.

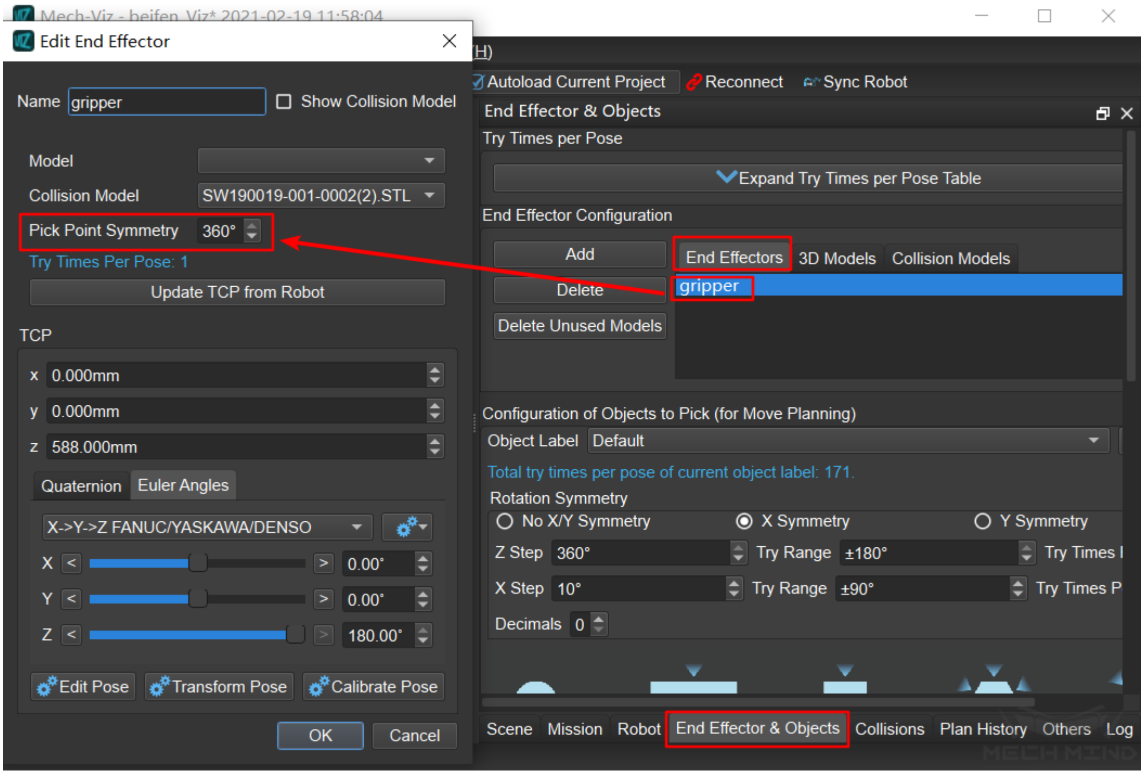
Attention: The distinction between the symmetry of the pick point and the picking relaxation directly depends on the output of the Mech-Vision project. Only when the Mech-Vision project uses map_to_multi_pick_points, the final effects of them are different. For Mech-Vision projects that only output poses, the picking relaxation is equivalent to the symmetry of the pick point.



Clarification on Symmetry and Picking Relaxation

1. **Pick Point Symmetry**

   The symmetry of the pick point is related to the way the end effector picks the object, and is used to select the optimal picking pose. The figure below shows how to set the symmetry of pick point:



Mech-Viz - beifen_Viz* 2021-02-19 11:58:04

Edit End Effector

Name gripper

□ Show Collision Model

Model

Collision Model SW190019-001-0002(2).STL

Pick Point Symmetry 360°

Try Times Per Pose: 1

TCP

x 0.000mm

y 0.000mm

z 588.000mm

X->Y->Z FANUC/YASKAWA/DENSO

X 0.00°

Y 0.00°

Z 180.00°

End Effector & Objects

Try Times per Pose

End Effector Configuration

End Effectors: gripper

Configuration of Objects to Pick (for Move Planning)

Object Label Default

Total try times per pose of current object label: 171.

Rotation Symmetry

○ No X/Y Symmetry ● X Symmetry ○ Y Symmetry

Z Step 360° Try Range ±180°

X Step 10° Try Range ±90°

Decimals 0

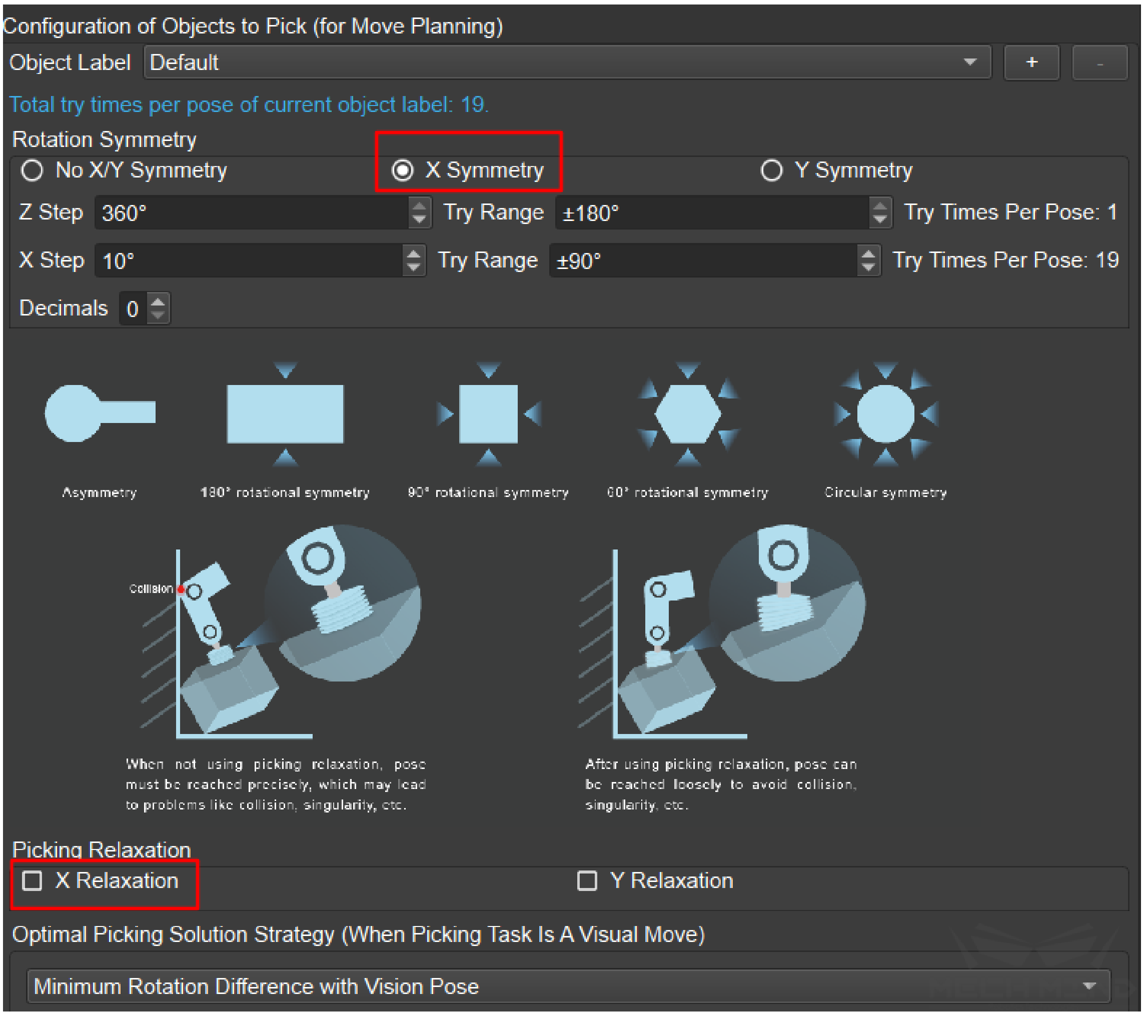
Please visit *Symmetry of Pick Point* for more information.

2. **Object Symmetry and Picking Relaxation**

   Object Label : When the visual point obtained from *visual_move* has a label, it will be searched here, and the symmetry settings of this type of object will be applied. When no matching label is found, the symmetry settings corresponding to the "Default" label will be applied. The "Default" label cannot be deleted, but user can add or delete other labels.

   Rotation Symmetry : In the X-axis symmetry and Y-axis symmetry of the object, only one of the two is supported (the Z-axis is not affected by this). This is because if an object has both X-axis and Y-axis symmetry, the object is generally a sphere, and this is usually not the case in practice. If user needs to set symmetry for a spherical object, please try to set it by picking relaxation instead of symmetry.

   Picking Relaxation : The picking relaxation supports the setting of X-axis relaxation and Y-axis relaxation at the same time, but it is applied separately during application. That is, the reference pose rotates around the X axis or the Y axis, instead of rotating around both axes at the same time, and there is a mutually exclusive relationship between the picking relaxation and the symmetry of the object. For example, X-axis relaxation cannot be selected if the X-axis symmetry is selected. As the figure below shows.



Configuration of Objects to Pick (for Move Planning)

Object Label: Default

Total try times per pose of current object label: 19.

Rotation Symmetry

○ No X/Y Symmetry ● X Symmetry ○ Y Symmetry

Z Step: 360° Try Range: ±180° Try Times Per Pose: 1

X Step: 10° Try Range: ±90° Try Times Per Pose: 19

Decimals: 0

When not using picking relaxation, pose must be reached precisely, which may lead to problems like collision, singularity, etc.

After using picking relaxation, pose can be reached loosely to avoid collision, singularity, etc.

Picking Relaxation

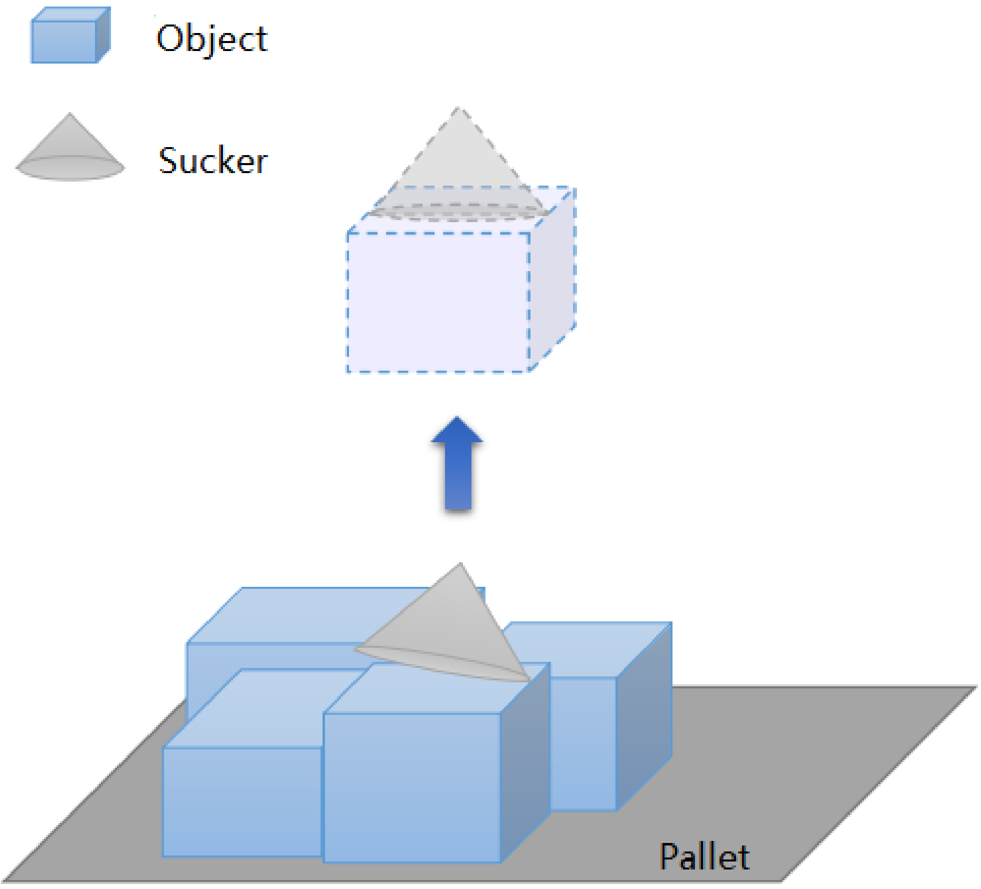
□ X Relaxation □ Y Relaxation

Optimal Picking Solution Strategy (When Picking Task Is A Visual Move)

Minimum Rotation Difference with Vision Pose

3. **The Difference between X/Y Axis Symmetry of Object and Picking Relaxation**

   - The application scenario of X/Y axis symmetry of object is that the relative position relationship between the end effector and the object on this axis is not allowed to change. Usually, X/Y axis symmetry is used when there are requirements for the placing pose of the object. For example, for crankshaft picking, the X-axis rotation angle of the crankshaft when it is finally placed is very important in many scenarios. At this time, it is necessary to set the X-axis symmetry so that the crankshaft can be placed exactly at the required angle.
   - The application scenario of the picking relaxation is that the relative positional relationship between the end effector and the object on the axis is allowed to change. For example, when using a sucker to pick object, even if there is a small angle deviation between the sucker and the object, but due to the flexibility of the sucker, the object can still be successfully picked, as shown in the following figure:

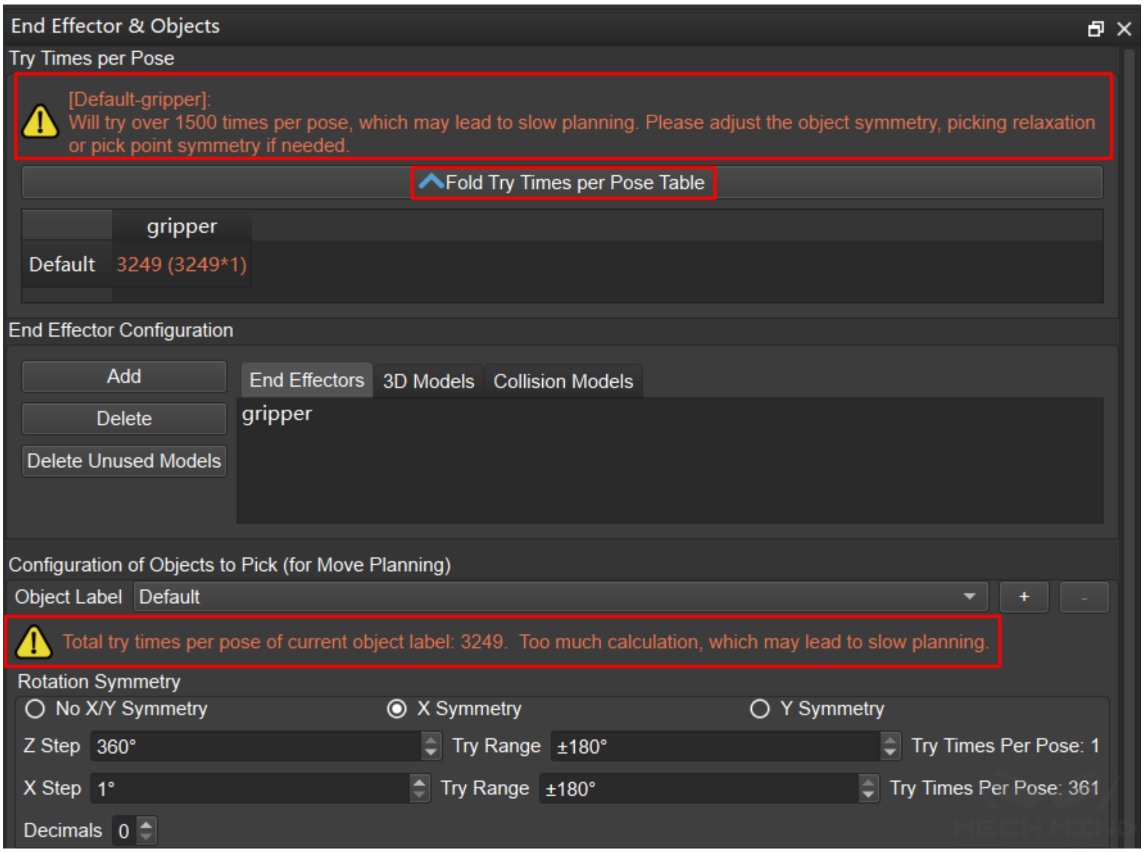


Pick Object with Sucker (small angle deviation)

The same is true for end effectors and workpieces. For scenes such as picking crankshafts, chain rail joints, etc., the object can still be picked if a small angular deviation is allowed in a certain axis direction of the object,.

4. **The planning time may be too long when the setting is not appropriate**

   When applying the symmetry of the pick point, the symmetry of the object, and the picking relaxation at the same time, since the number of attempts is the product of the respective number of values of these three items, improper settings may result in too many attempts and the planning time may be too long. Mech-Viz may even crash. Therefore, the software will give a warning when the number of attempts exceeds 1500. The warning can be viewed in detail by clicking *Expand Try Times per Pose Table*, as shown in the figure below:



End Effector & Objects

Try Times per Pose

[Default-gripper]: Will try over 1500 times per pose, which may lead to slow planning. Please adjust the object symmetry, picking relaxation or pick point symmetry if needed.

Fold Try Times per Pose Table

	gripper
Default	3249 (3249*1)

End Effector Configuration

Add

Delete

Delete Unused Models

End Effectors | 3D Models | Collision Models

gripper

Configuration of Objects to Pick (for Move Planning)

Object Label: Default

Total try times per pose of current object label: 3249. Too much calculation, which may lead to slow planning.

Rotation Symmetry

○ No X/Y Symmetry ● X Symmetry ○ Y Symmetry

Z Step 360° Try Range ±180° Try Times Per Pose: 1

X Step 1° Try Range ±180° Try Times Per Pose: 361

Decimals 0

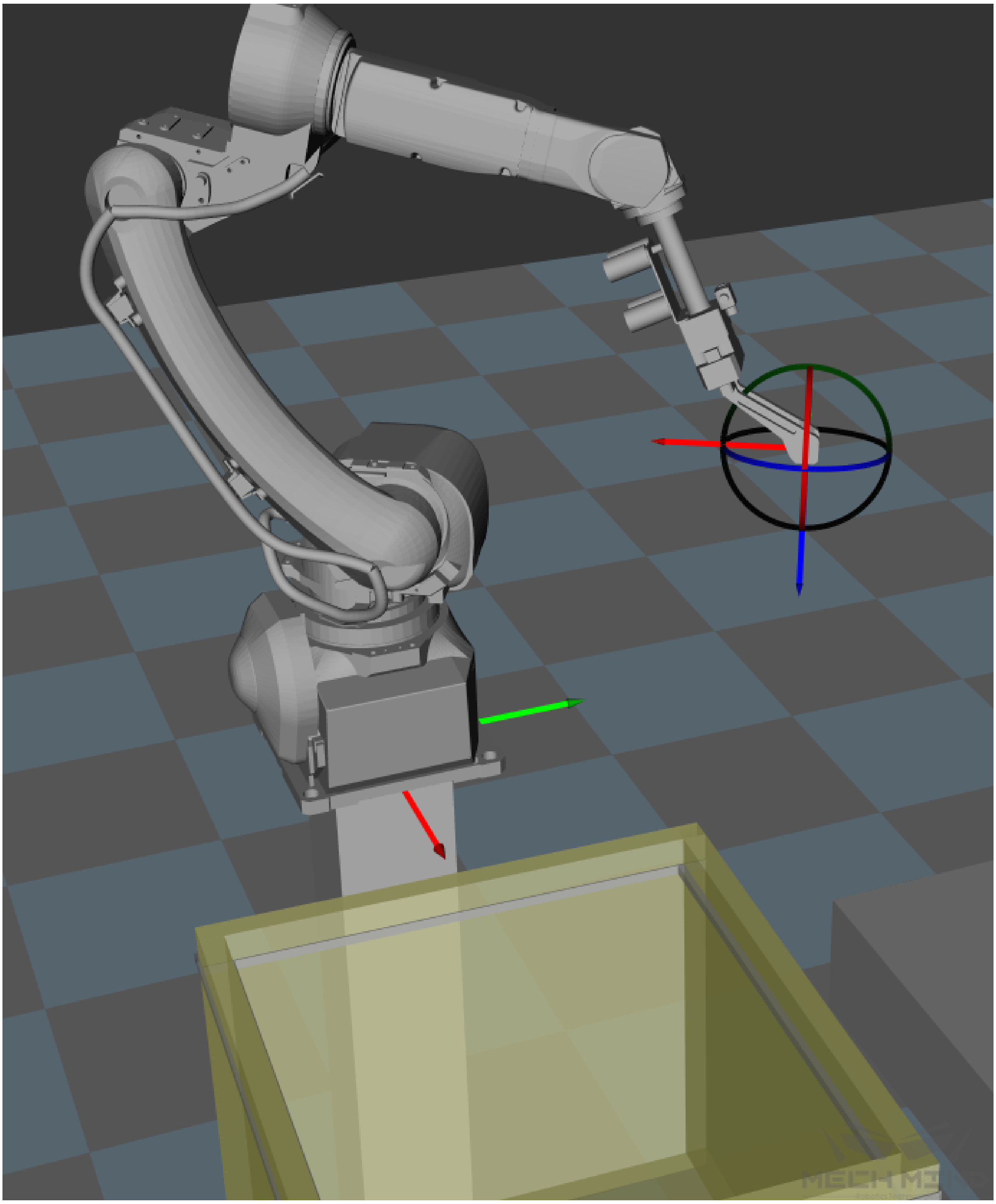
Optimal Picking Solution Strategy (When Picking Task Is A Visual Move)

There are currently three modes: Default, Minimum Global Rotation, and Minimum Rotation Difference with Vision Pose.

For compatibility with functions that may be developed in the future, the “Default” mode is introduced. For the current version, “Default” is equivalent to “Minimum Global Rotation”.

Minimum Global Rotation: When this mode is selected, symmetry will be applied according to the principle of the minimum Z-axis rotation of the pick points in the whole process of “pick-place”. The advantage of this mode is that it can avoid the robot from turning meaninglessly after picking the object, and avoiding the risk of the picked object falling.

Minimum Rotation Difference with Vision Pose: this mode is introduced because in some special application scenarios, the rotation during the picking and placing process does not matter, and the end effector needs to be visually guided to rotate to achieve the goal. For example, the end effector of a certain project is shown in the figure below. The object is a cylinder and is symmetric at any angle. Therefore, the end effector can pick the object at any Z-axis angle as long as it does not cause interference. When picking objects close to the wall of the box in a deep box, if “Minimum Global Rotation” is applied, the picking pose may be that the end effector is close to the box wall for picking, which is high risky. At this time, the Z-axis angle of picking pose can be guided by vision, such as editing the Mech-Vision project so that the X-axis of the object poses all point to the center of the box. In this way, when planning the picking pose, keeping the rotation difference with the visual pose to a minimum can guide the end effector to be as far away as possible from the box wall when picking, and improve the safety of picking. This Mech-Vision project gives two kinds of labels, which represent the center and the inner wall of the box respectively. Edit these two types of labels separately in Mech-Viz. Apply “Minimum Global Rotation” to the objects in the center of the box to avoid unnecessary rotation as much as possible; Apply “Minimum Rotation Difference with Vision Pose” to the objects around inner wall of the box, and Mech-Vision guides Mech-Viz to select a safer picking pose.



Object Size to Pick (When Picking Task Is Not A Visual Move)

It has nothing to do with the object label. It is a global parameter. It is used to give a size to the picked object when picking task is not a visual move. Generally used for testing by developers.

2.3.5 Collision Detection

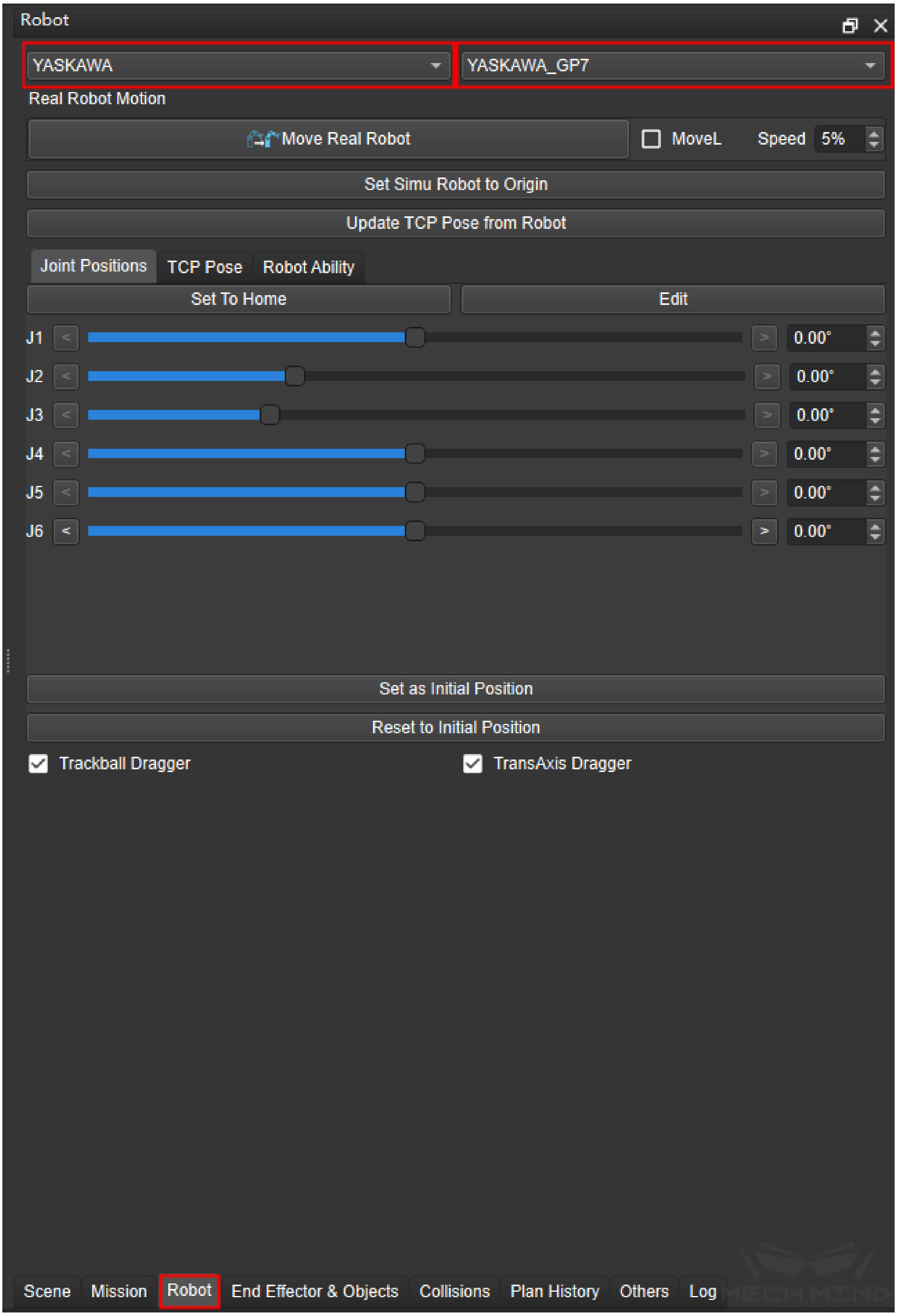
This chapter mainly introduces collision detection related to settings and precautions. Setting collision detection in Mech-Viz is mainly divided into two major steps: configuring the collision model and configuring the collision detection parameters.

Collision Model Configuration

Collision models in Mech-Viz include robot links, scene objects, end effector, point cloud, detected objects, picked objects and placed objects. Each collision model type is different so the calculation and method of detection will also be different. For the distinction of collision model types, please refer to *Collision Model Type* .

The configuration methods of each collision model are different, the specific configuration method of each model is as follows :

Robot Links: Select *Robot* at the buttom on the right, then select the brand of the robot and the corresponding robot model, as shown in the figure below.



YASKAWA

YASKAWA_GP7

□ MoveL

Speed 5%

J1 0.00°

J2 0.00°

J3 0.00°

J4 0.00°

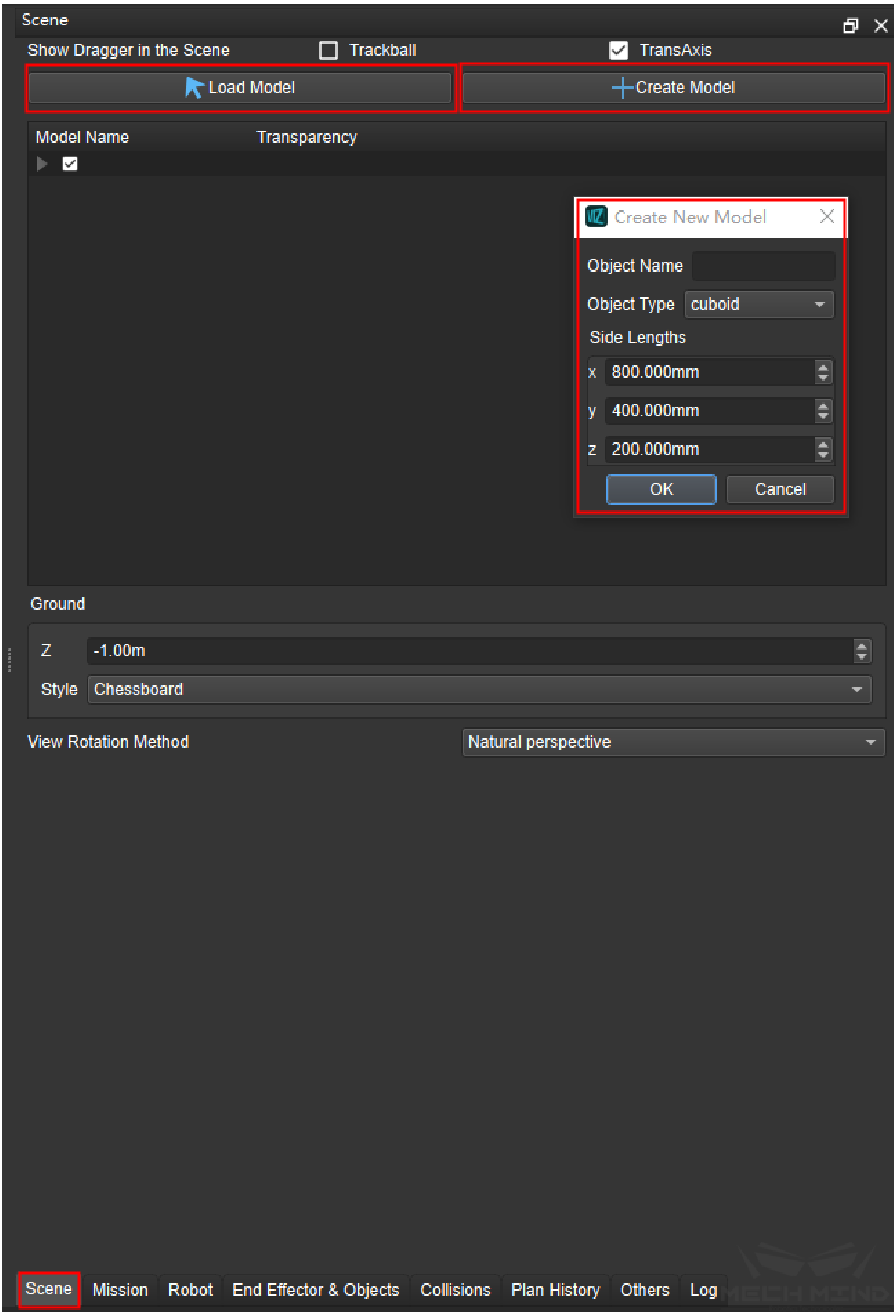
J5 0.00°

J6 0.00°

☑ Trackball Dragger

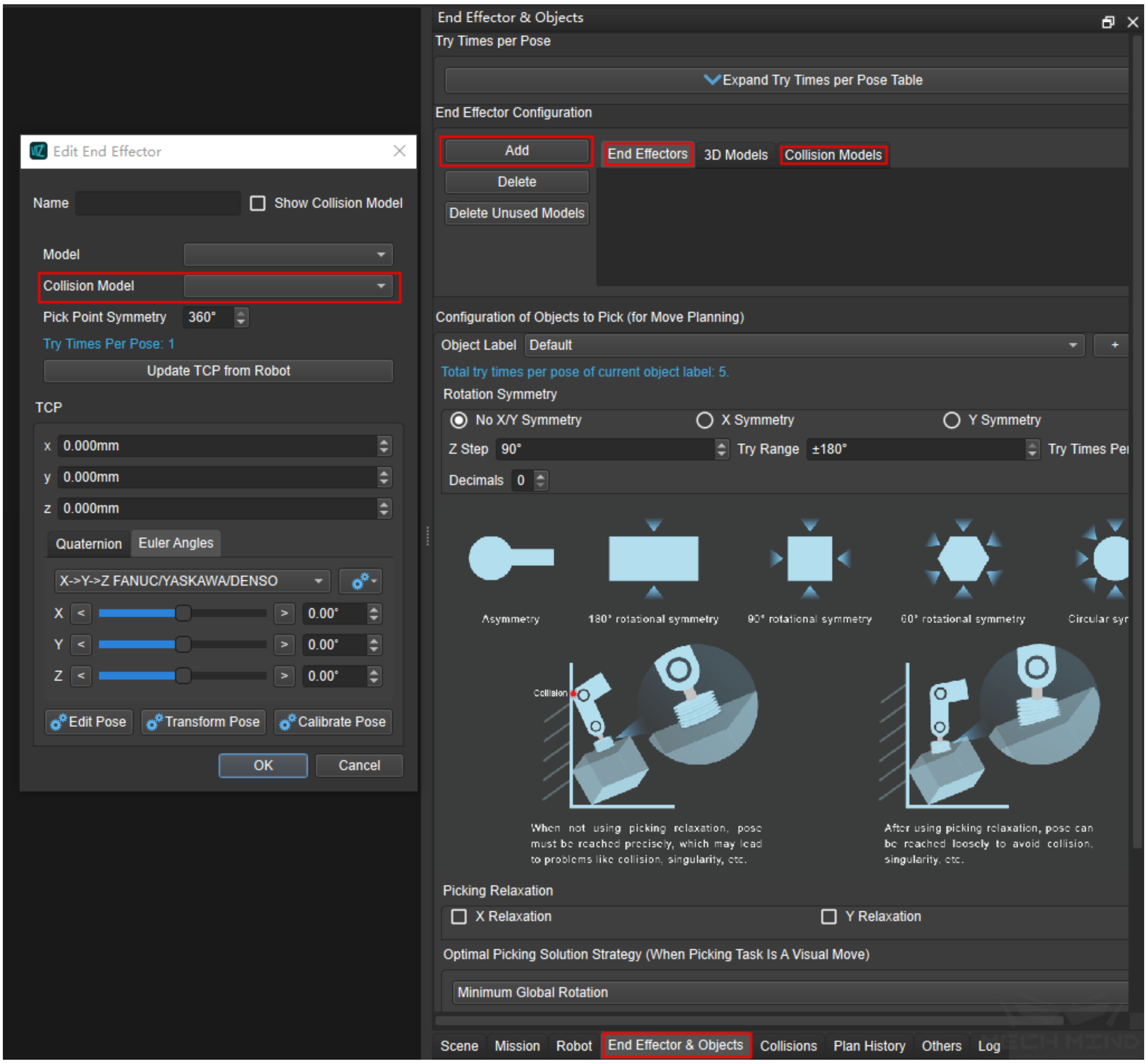
☑ TransAxis Dragger

Scene Objects: Select *Scene* at the buttom on the right, then click *Create Model* to create cuboid or box (selectd by object type) , you can also click *Load Model* to load the existing scene object model (support stl, obj, dae and other foramts), as shown in the figure below.


Scene
Show Dragger in the Scene
□ Trackball
☑ TransAxis
Load Model
Create Model
Model Name
Transparency
Create New Model
Object Name
Object Type
cuboid
Side Lengths
x
800.000mm
y
400.000mm
z
200.000mm
OK
Cancel
Ground
Z
-1.00m
Style
Chessboard
View Rotation Method
Natural perspective
Scene
Mission
Robot
End Effector & Objects
Collisions
Plan History
Others
Log

> **Attention:** The model imported by loading needs to be checked the *Need Collision Detection* in the additional parameter before it can be actually added to the collision detection, ortherwise it can only be displayed as a 3D model.

End Effector: Select *End Effector & Objects* at the buttom on the right, then select the collision model and click *Add* on the left to select the tool that needs to be added. At this time, the collision model does not actually take effect, you need to select the collision model in the end effector, as shown the figure below.



Pick Point Symmetry 360°

Try Times Per Pose: 1

x 0.000mm

y 0.000mm

z 0.000mm

X->Y->Z FANUC/YASKAWA/DENSO

X 0.00°

Y 0.00°

Z 0.00°

Object Label Default

Total try times per pose of current object label: 5.

Z Step 90° Try Range ±180°

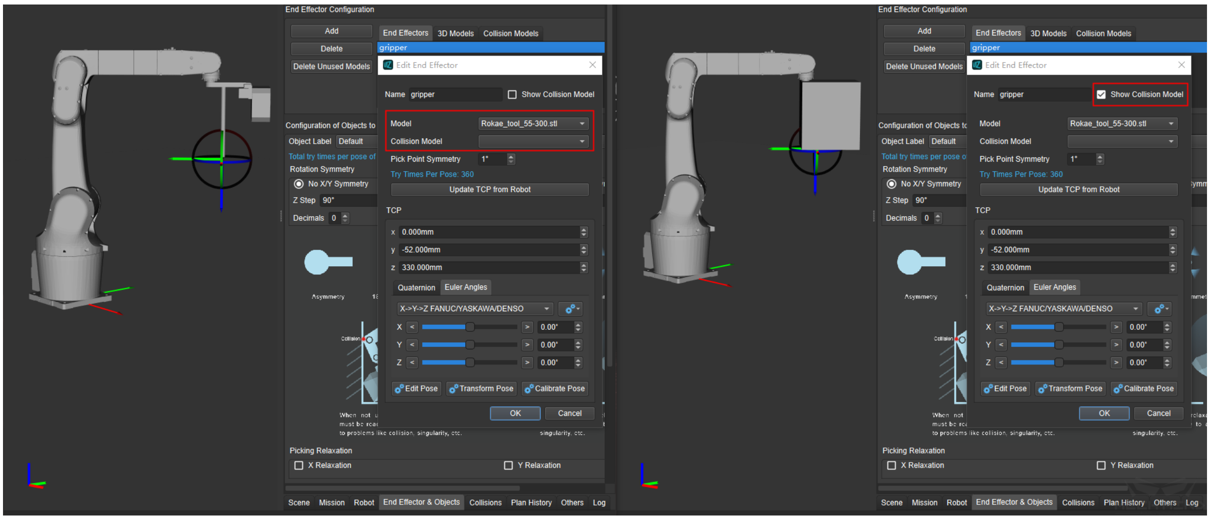
Decimals 0

When not using picking relaxation, pose must be reached precisely, which may lead to problems like collision, singularity, etc.

After using picking relaxation, pose can be reached loosely to avoid collision, singularity, etc.

Minimum Global Rotation

Tip: If only the 3D model is available, the bounding box (basic geometry type) can be used as a collision model for the collision detection. You just need to click *Show Collision Model* after importing the 3D model in the end effector, as shown in the figure below.



End Effector Configuration

Name: gripper — □ Show Collision Model (left); ☑ Show Collision Model (right)

Model: Rokae_tool_55-300.stl

Collision Model:

Pick Point Symmetry: 1°

Try Times Per Pose: 360

TCP

x 0.000mm

y -52.000mm

z 330.000mm

X->Y->Z FANUC/YASKAWA/DENSO

X 0.00°

Y 0.00°

Z 0.00°

Configuration of Objects to

Object Label: Default

Rotation Symmetry: ● No X/Y Symmetry

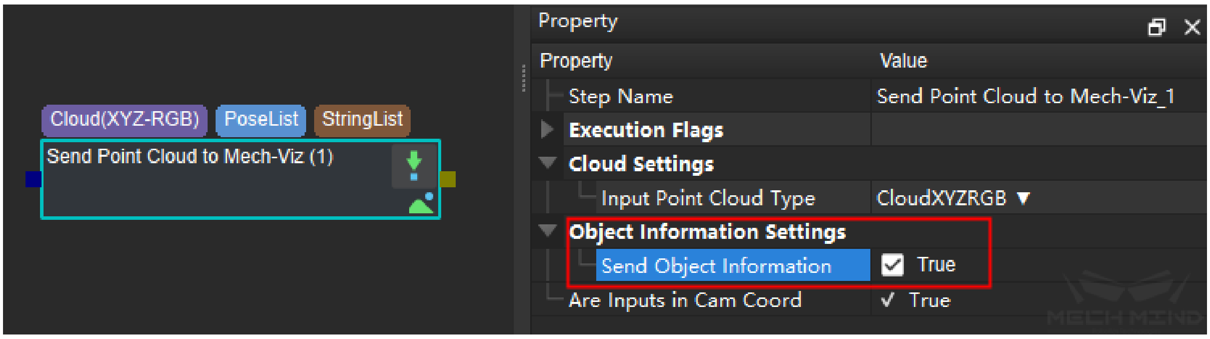
Z Step: 90°

Decimals: 0

Picking Relaxation: □ X Relaxation □ Y Relaxation

Point Cloud: The point cloud model are mostly the detected objects in the process, it need to be imported through the step send_point_cloud_to_external_service in Mech-Vision.

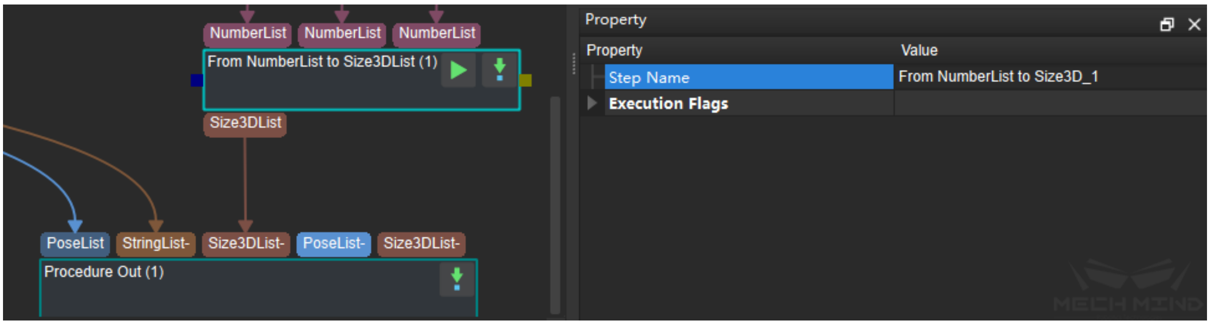
Detected Objects: This kind of model need to add the stl model and binvox model of the objects to the collision_models folder in the project, and then use the step send_point_cloud_to_external_service through Mech-Vision to send objects information (including pose, label and point cloud etc.) to Mech-Viz by checking *Send Object Information* as Ture in the parameter settings. For more information about the conversion of the binvox model, please refer to *Collision Models of .binvox Format* .



Property

Property	Value
Step Name	Send Point Cloud to Mech-Viz_1
Execution Flags	
Cloud Settings	
Input Point Cloud Type	CloudXYZRGB ▼
Object Information Settings	
Send Object Information	☑ True
Are Inputs in Cam Coord	✓ True

Picked Objects: This kind of model is mainly divided into two categories: cuboid and non-cuboid. The configuration method of the non-cuboid model is the same as the detected objects model, the cuboid model need to use the step “From NumberList to Size3DList” to calculate the object size and then send the result to Mech-Viz through the step procedure_out, after receiving the result Mech-Viz will generate the corresponding collision model, as shown in the figure below.



Property

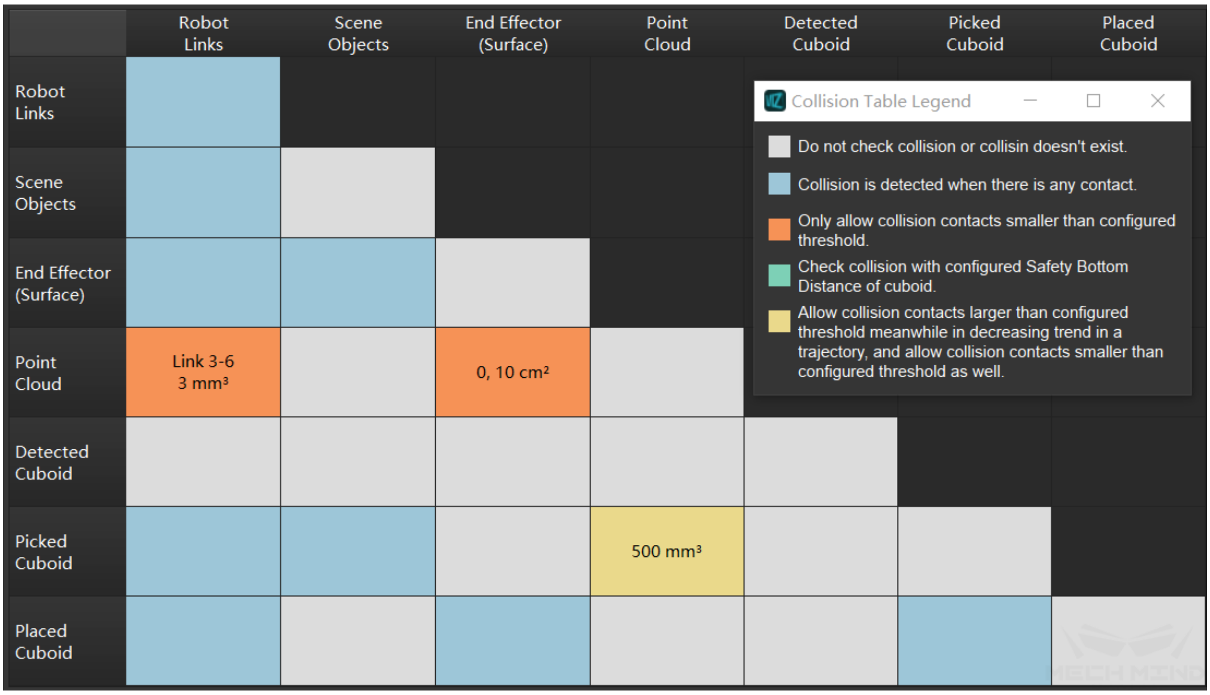
Property	Value
Step Name	From NumberList to Size3D_1
Execution Flags	

Placed Objects: This kind of model does not need to be configured in most cases. The placed object model is configured by using the picked object model or the detected object model. The collision detection of this kind of model belong to scene collision.

Collision Detection Parameters Configuration

> **Attention:** It is required to configure the corresponding collision model before configuring the collision detection parameters, otherwise the collision detection cannot be performed or error messages will appear.

In Mech-Viz, collision detection is performed by detecting the collisions between every two collision models. Therefore, there are multiple types of collision detection combinations, such as collisions between point clouds and other objects, collisions between robots and other objects, etc. In Mech-Viz, you can use the collision table to learn the type of collision detection combination performed by the current project. Different colors in the table represent different collision detection conditions, as shown in the figure below.



	Robot Links	Scene Objects	End Effector (Surface)	Point Cloud	Detected Cuboid	Picked Cuboid	Placed Cuboid
Robot Links	🟦						
Scene Objects	🟦	⬜					
End Effector (Surface)	🟦	🟦	⬜				
Point Cloud	🟧 Link 3-6 3 mm^3	⬜	🟧 0, 10 cm^2	⬜			
Detected Cuboid	⬜	⬜	⬜	⬜	⬜		
Picked Cuboid	🟦	🟦	⬜	🟨 500 mm^3	⬜	⬜	
Placed Cuboid	🟦	⬜	🟦	⬜	⬜	🟦	⬜

Collision Table Legend

⬜ Do not check collision or collisin doesn't exist.

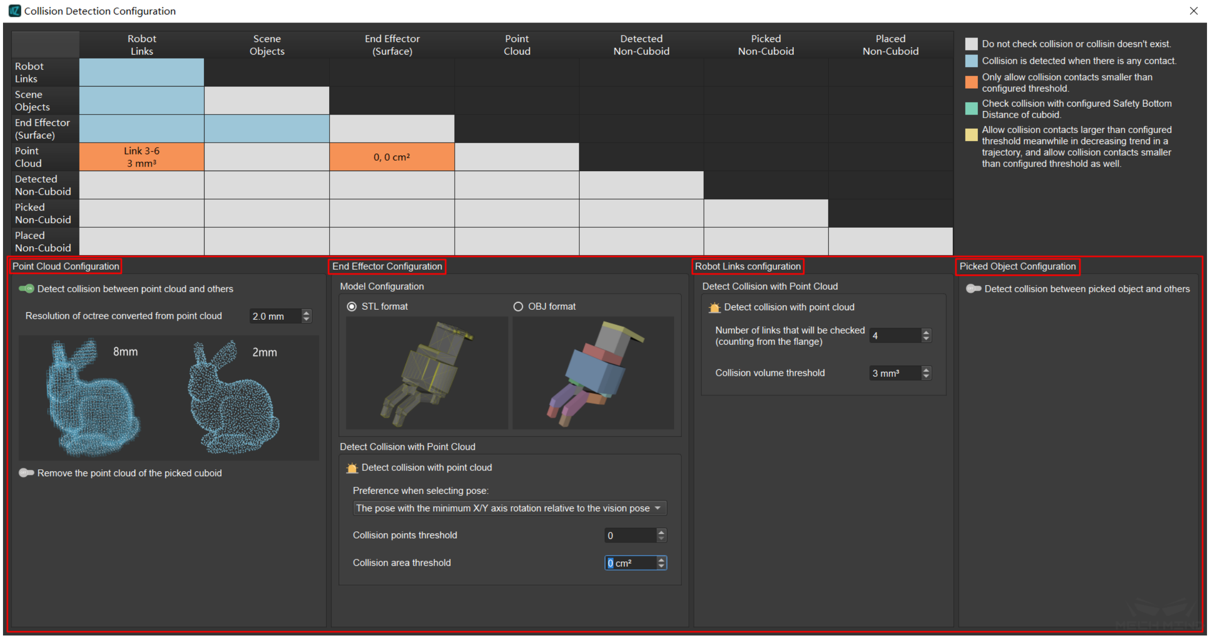
🟦 Collision is detected when there is any contact.

🟧 Only allow collision contacts smaller than configured threshold.

🟩 Check collision with configured Safety Bottom Distance of cuboid.

🟨 Allow collision contacts larger than configured threshold meanwhile in decreasing trend in a trajectory, and allow collision contacts smaller than configured threshold as well.

In the collision detection configuration (globally) we have the following four configurations: point cloud configuration, end effector configuration, robot links configuration and picked object configuration, as shown in the following figure. Proper configuration of collision detection can remove unnecessary and redundant collision detection and improve efficiency.



Collision Detection Configuration

	Robot Links	Scene Objects	End Effector (Surface)	Point Cloud	Detected Non-Cuboid	Picked Non-Cuboid	Placed Non-Cuboid
Robot Links	🟦						
Scene Objects	🟦	⬜					
End Effector (Surface)	🟦	🟦	⬜				
Point Cloud	🟧 Link 3-6 3 mm³	⬜	🟧 0, 0 cm²	⬜			
Detected Non-Cuboid	⬜	⬜	⬜	⬜	⬜		
Picked Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	
Placed Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	⬜

⬜ Do not check collision or collisin doesn't exist.

🟦 Collision is detected when there is any contact.

🟧 Only allow collision contacts smaller than configured threshold.

🟩 Check collision with configured Safety Bottom Distance of cuboid.

🟨 Allow collision contacts larger than configured threshold meanwhile in decreasing trend in a trajectory, and allow collision contacts smaller than configured threshold as well.

Point Cloud Configuration

Detect collision between point cloud and others

Resolution of octree converted from point cloud: 2.0 mm

Remove the point cloud of the picked cuboid

End Effector Configuration

Model Configuration

● STL format ○ OBJ format

Detect Collision with Point Cloud

Detect collision with point cloud

Preference when selecting pose:

The pose with the minimum X/Y axis rotation relative to the vision pose

Collision points threshold: 0

Collision area threshold: 0 cm²

Robot Links configuration

Detect Collision with Point Cloud

Detect collision with point cloud

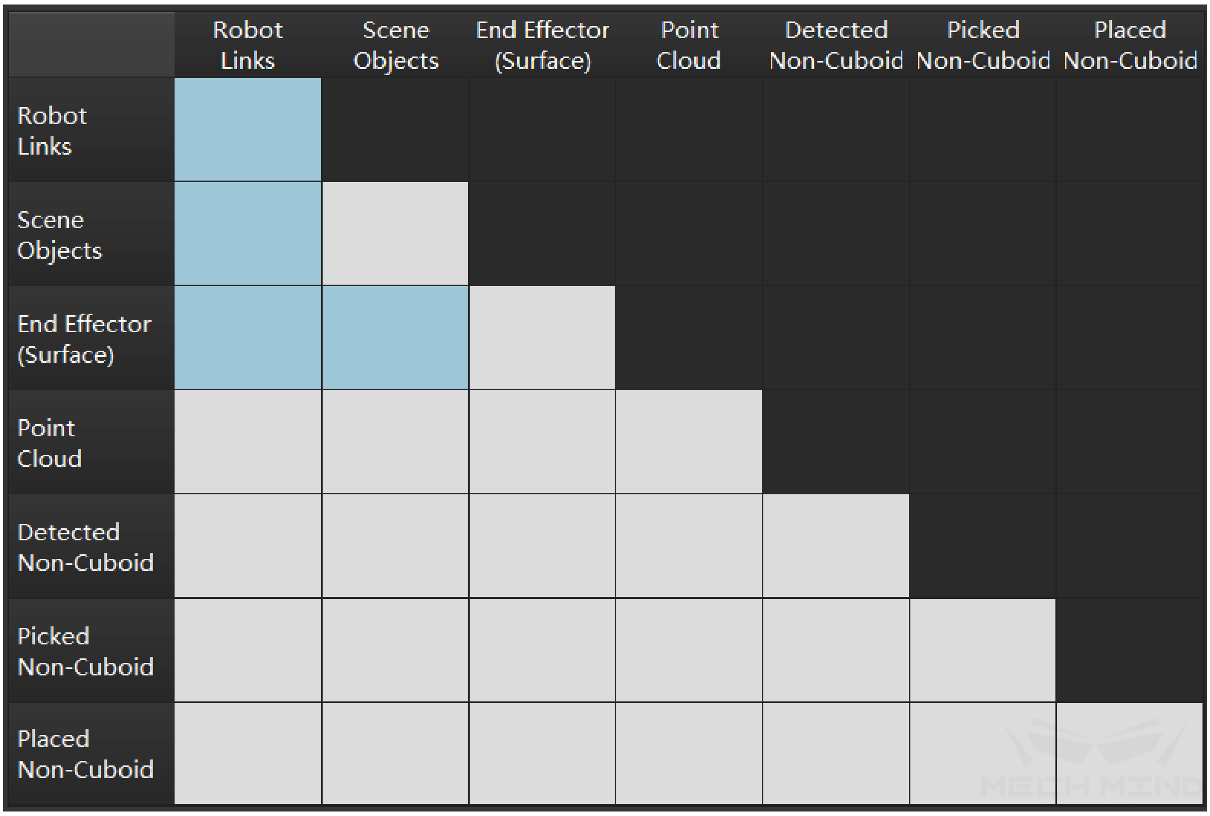
Number of links that will be checked (counting from the flange): 4

Collision volume threshold: 3 mm³

Picked Object Configuration

Detect collision between picked object and others

Without any of these four collision detection configurations, only collisions between robot links, scene objects and end effector are performed. This is the default collision detection configuration, and the collision detection performed at this time is scene collision and self collision, as shown in the figure below.



	Robot Links	Scene Objects	End Effector (Surface)	Point Cloud	Detected Non-Cuboid	Picked Non-Cuboid	Placed Non-Cuboid
Robot Links	🟦						
Scene Objects	🟦	⬜					
End Effector (Surface)	🟦	🟦	⬜				
Point Cloud	⬜	⬜	⬜	⬜			
Detected Non-Cuboid	⬜	⬜	⬜	⬜	⬜		
Picked Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	
Placed Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	⬜

Point Cloud Configuration

If point cloud related collision detection is needed in the project, please turn on the point cloud configuration and set the parameters. The configuring procedure are as follows:

Step 1. Use to enable or disable point cloud collision detection. When using point cloud collision detection, please make sure that the step “Send Point Cloud to Mech-Viz” is used in Mech-Vision.

Step 2. Set the resolution of octree collision model converted from point cloud. The resolution here refers to the side length of each voxel when the point cloud is transformed into voxels. For the same point cloud, the lower the point cloud resolution, the more the number of voxel cubes, the more accurate the calculation and the longer the time; on the contrary, the larger the resolution, the less the number of voxel cubes, the calculation is relatively rough but the speed is faster.

Attention: When performing point cloud collision detection, what it detects is actually the collision between the end effector and the voxel cube, not the point in the point cloud.

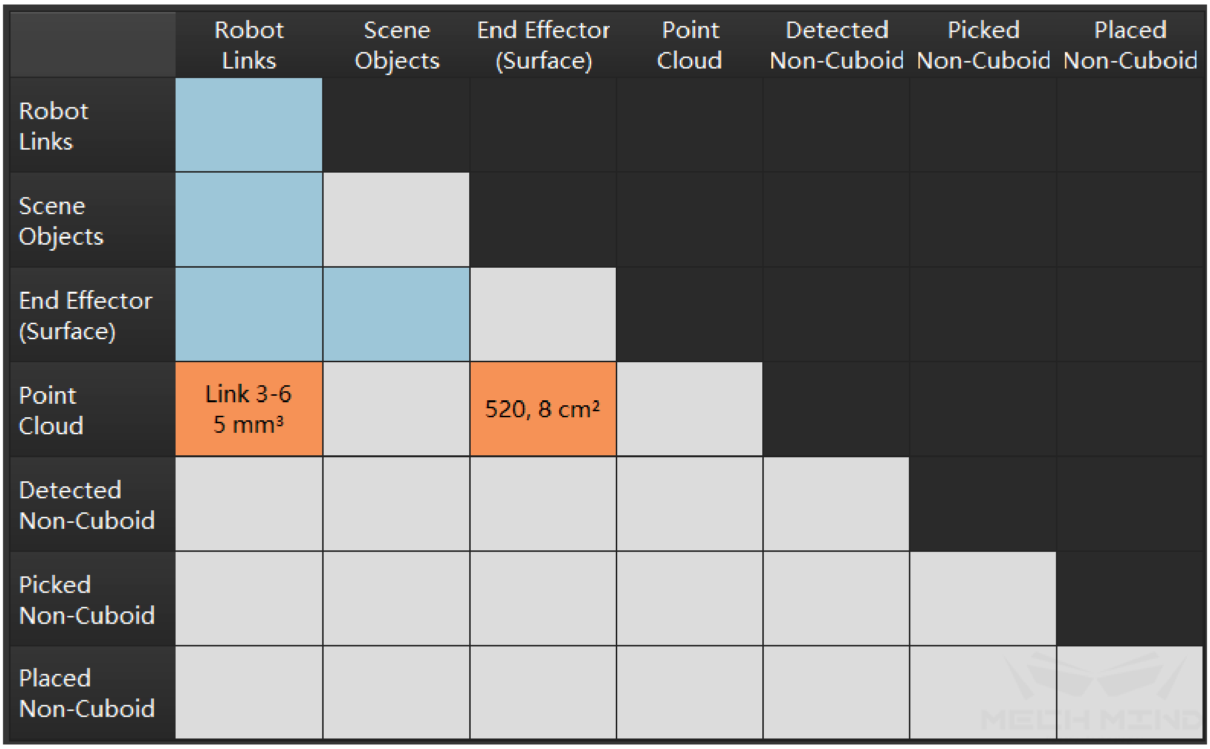
Step 3. Use to choose whether to remove the point cloud of the picked cuboid.

Step 4. Set the parameters of *Additional removed length along X/Y axis* and *Additional removed height along positive Z axis*. The length here refers to the additional removal of the point cloud within the length range when removing the point cloud.

Tip:

1. *Additional removed length along X/Y axis* can not be set too large, otherwise it is easy to remove the surrounding point cloud that needs collision detection.
2. *Additional removed height along positive Z axis* can be relatively large to ensure that anomalies caused by point cloud fluctuations can be removed.

After configuration, visual collision detection is introduced. At this time, the collision detection is scene collision, self collision, and point cloud collision, as shown in the following figure.



	Robot Links	Scene Objects	End Effector (Surface)	Point Cloud	Detected Non-Cuboid	Picked Non-Cuboid	Placed Non-Cuboid
Robot Links	🟦						
Scene Objects	🟦	⬜					
End Effector (Surface)	🟦	🟦	⬜				
Point Cloud	Link 3-6 5 mm³	⬜	520, 8 cm²	⬜			
Detected Non-Cuboid	⬜	⬜	⬜	⬜	⬜		
Picked Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	
Placed Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	⬜

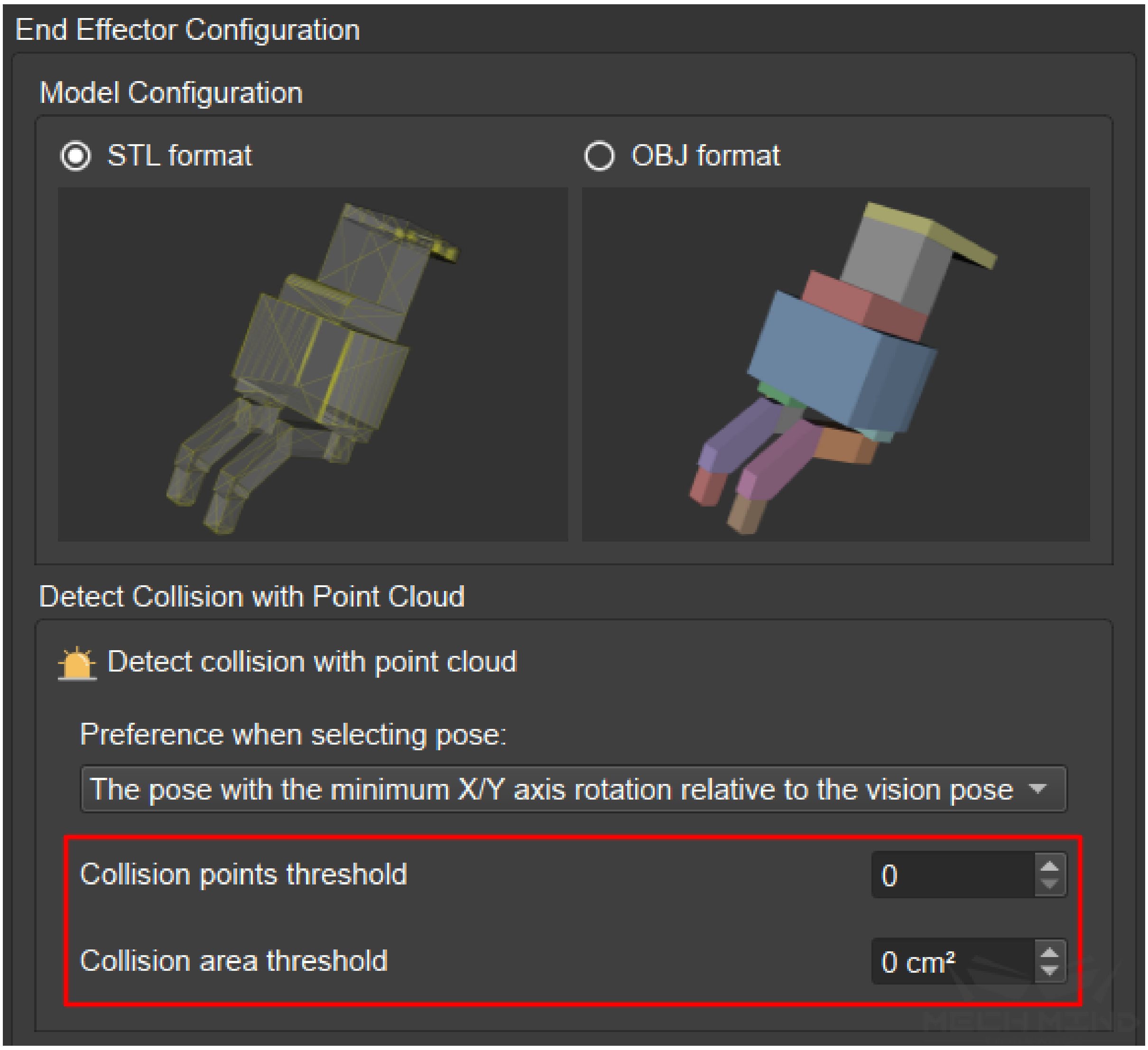
The above is the method of globally configuring point cloud collision detection. In Mech-Viz, you can also configure point cloud collision detection separately for some movement related Skills. For details, please visit *General Parameters of Move Skills*.

End Effector Configuration

The collision detection of the end effector is always turned on by default. The configuration of the end effector is mainly to select the tool models in two different formats of stl and obj and the corresponding parameter settings.

The main difference between the two formats is that the stl format only describes the surface geometry of the three-dimensional object, so Mech-Viz only detects the collision between the model surface and the point cloud; while the obj format is used as a combination of multiple solid convex bodies, Mech-Viz will detect both the model surface and internal collisions with the point cloud.

- **stl format** *Collision points threshold* and *Collision area threshold* should be set, as shown in the figure below.



End Effector Configuration

Model Configuration

● STL format ○ OBJ format

Detect Collision with Point Cloud

Detect collision with point cloud

Preference when selecting pose:

The pose with the minimum X/Y axis rotation relative to the vision pose

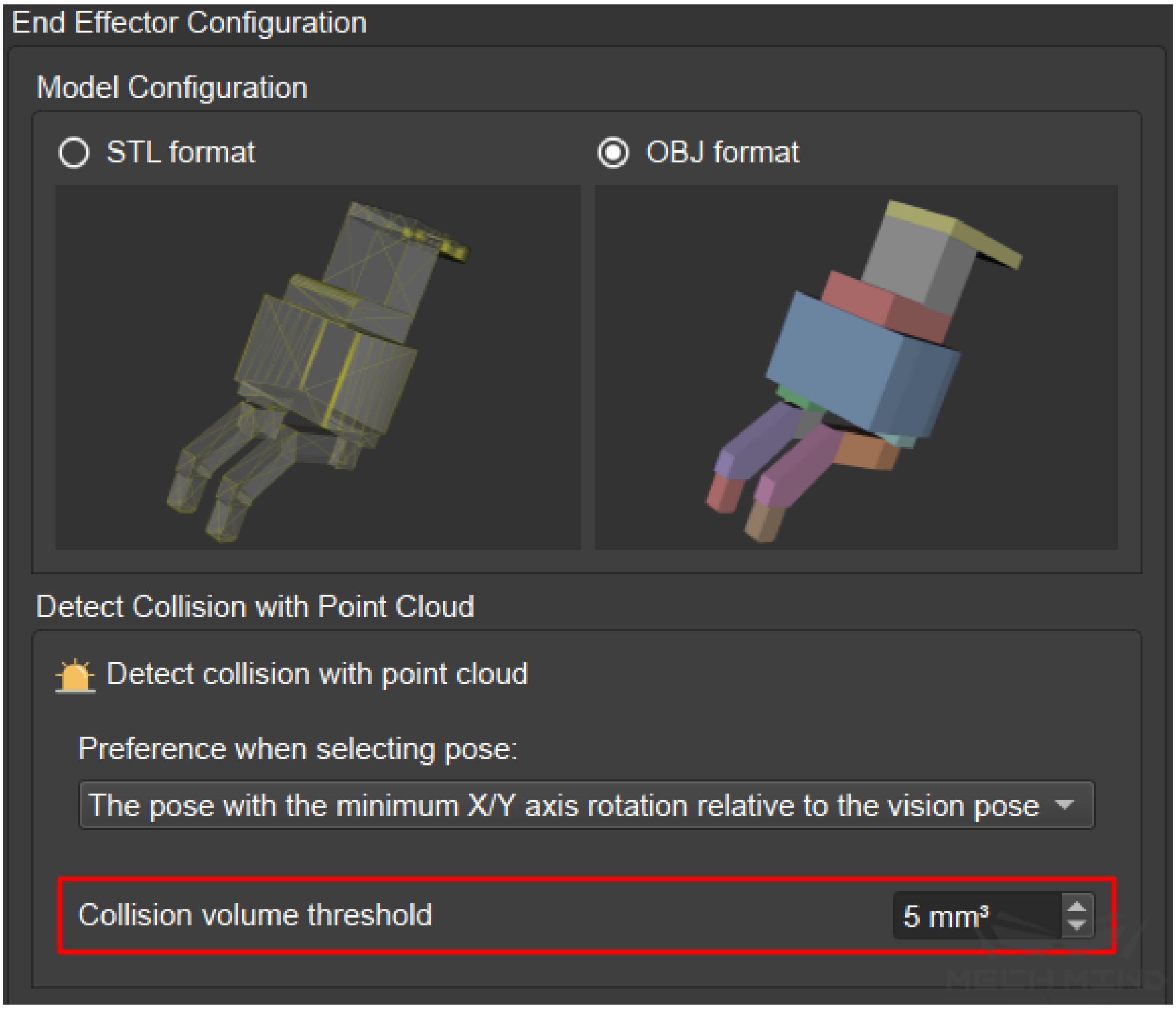
Collision points threshold: 0

Collision area threshold: 0 cm²

Collision points threshold refers to the number of points that are allowed to collide with the surface of the model when picking.

Collision area threshold refers to the area that allows the point cloud to sweep across the surface of the model in a trajectory.

- **obj format** Only *Collision volume threshold* need to be set, as shown in the figure below.



End Effector Configuration

Model Configuration

○ STL format ● OBJ format

Detect Collision with Point Cloud

Detect collision with point cloud

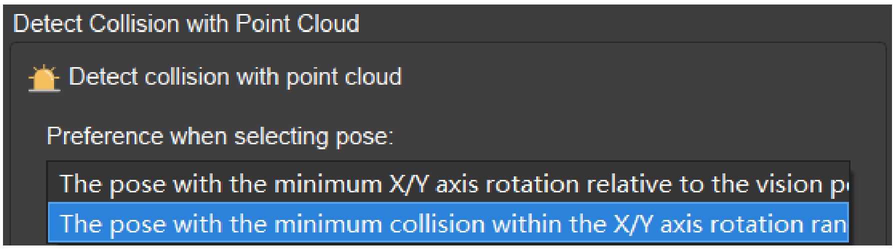
Preference when selecting pose:

The pose with the minimum X/Y axis rotation relative to the vision pose

Collision volume threshold 5 mm³

Collision volume threshold refers to the allowed collision volume between the end effector and the point cloud during collision detection.

Preference when selecting pose is a parameter that needs to be set for both formats. It has two options, which are `The pose with the minimum X/Y axis rotation relative to the vision pose` and `The pose with the minimum collision within the X/Y axis rotation range`. as shown in the figure below.



Detect Collision with Point Cloud

Detect collision with point cloud

Preference when selecting pose:

The pose with the minimum X/Y axis rotation relative to the vision p

The pose with the minimum collision within the X/Y axis rotation ran

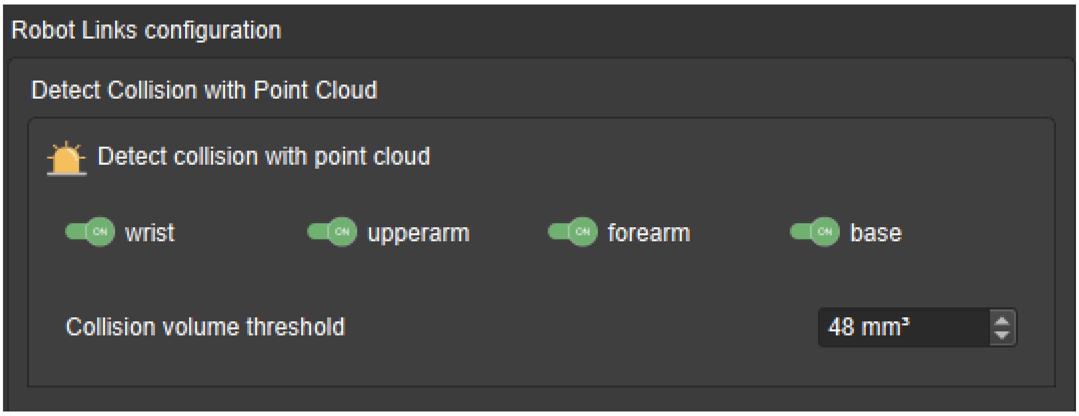
`The pose with the minimum X/Y axis rotation relative to the vision pose` means that take the sorted pose obtained from the visual movement as the given pose, and check the pose with the smallest X/Y axis rotation relative to the given pose.

`The pose with the minimum collision within the X/Y axis rotation range` means that calculate all point cloud collisions of the X/Y axis symmetry, and then sort the collisions from minimum to maximum.

> **Attention:** The above two parameters are premised on the existence of symmetry in pose.

Robot Links Configuration

The collision detection of the robot links is on by default. The Robot Links Configuration interface sets the parameters for detecting the collision between the robot links and the point cloud, as shown in the following figure.



Robot Links configuration

Detect Collision with Point Cloud

Detect collision with point cloud

wrist: ON

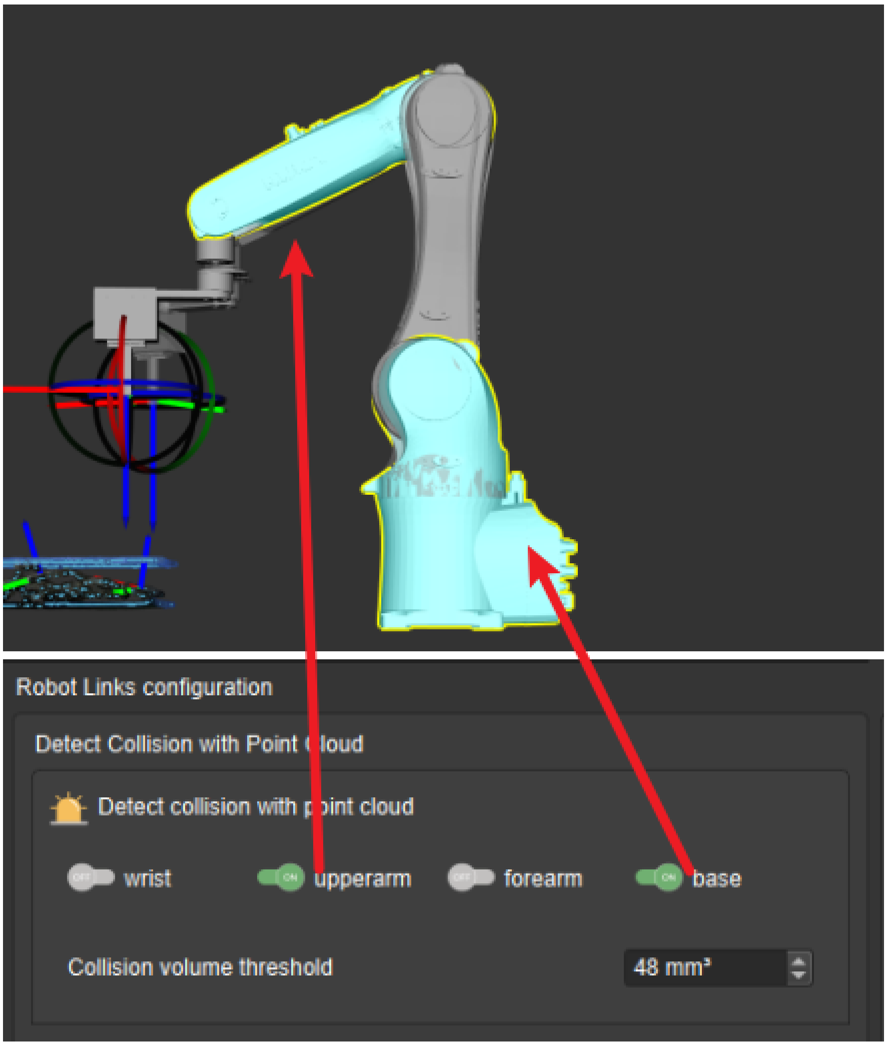
upperarm: ON

forearm: ON

base: ON

Collision volume threshold: 48 mm³

- The robot components that may be involved in point cloud collision detection include the wrist, the upper-arm, the fore-arm, and the base. In the Collision Detection Configuration interface, the component(s) that need to be involved in collision detection can be selected. The component(s) selected will be highlighted.


Robot Links configuration

Detect Collision with Point Cloud

Detect collision with point cloud

OFF wrist   ON upperarm   OFF forearm   ON base

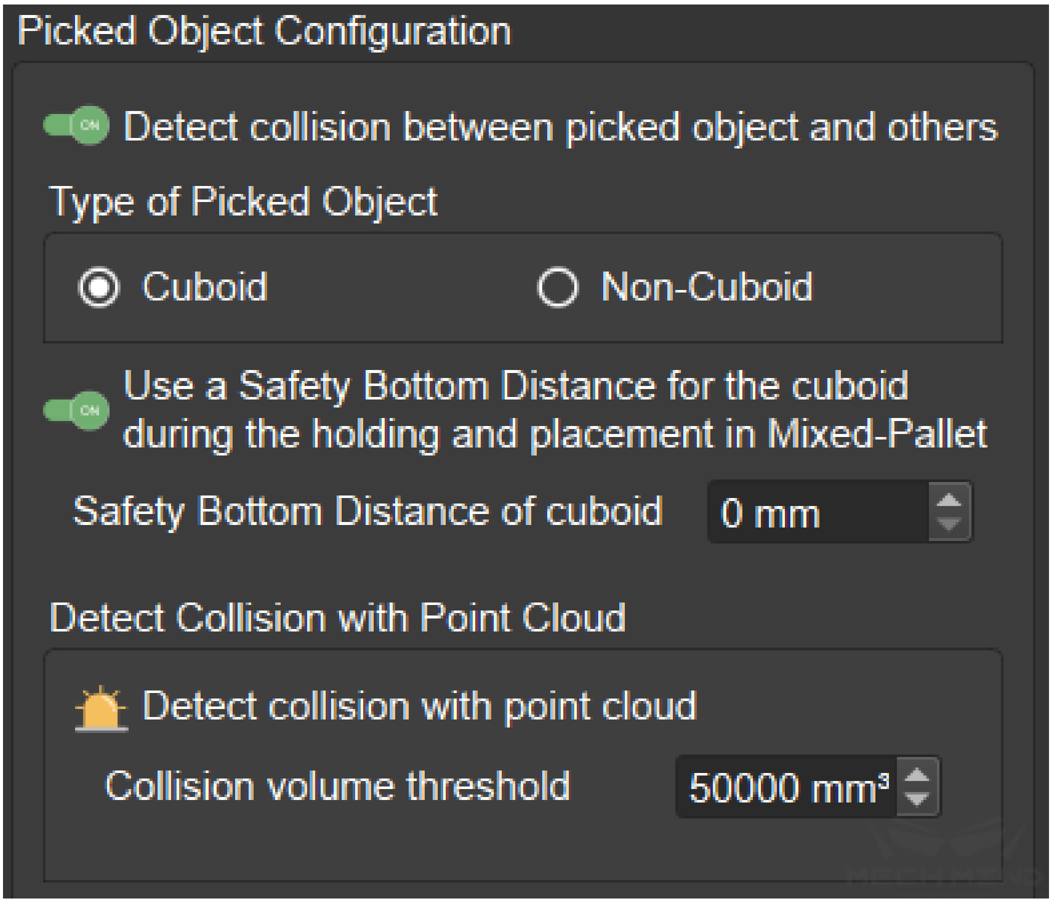
Collision volume threshold: 48 mm³

- *Collision volume threshold* refers to the allowed collision volume between the robot and the point cloud when performing collision detection. However, in the simulation, the robot is an STL format model, which means only the collisions between the surface and the point cloud can be detected, so the collision volume here refers to the product of the number of voxels that collide with the robot model surface and the volume of a voxel.

Picked Object Configuration

In the collision detection, the types of picked objects are mainly divided into two categories: cuboid and non-cuboid. The parameters that need to be configured of the two types are roughly the same, and the main difference is the models and the methods of importing. For detailed information, please visit *Collision Model Configuration*.

- **Cuboid**: When detecting the collision with the point cloud, since the box is a basic geometry, a complete collision can be detected. At this time, the detection content is that the collision between the picked cuboid and the point cloud, and the parameter that needs to be set is collision volume threshold, as shown in the figure below.



Picked Object Configuration

ON Detect collision between picked object and others

Type of Picked Object

● Cuboid ○ Non-Cuboid

ON Use a Safety Bottom Distance for the cuboid during the holding and placement in Mixed-Pallet

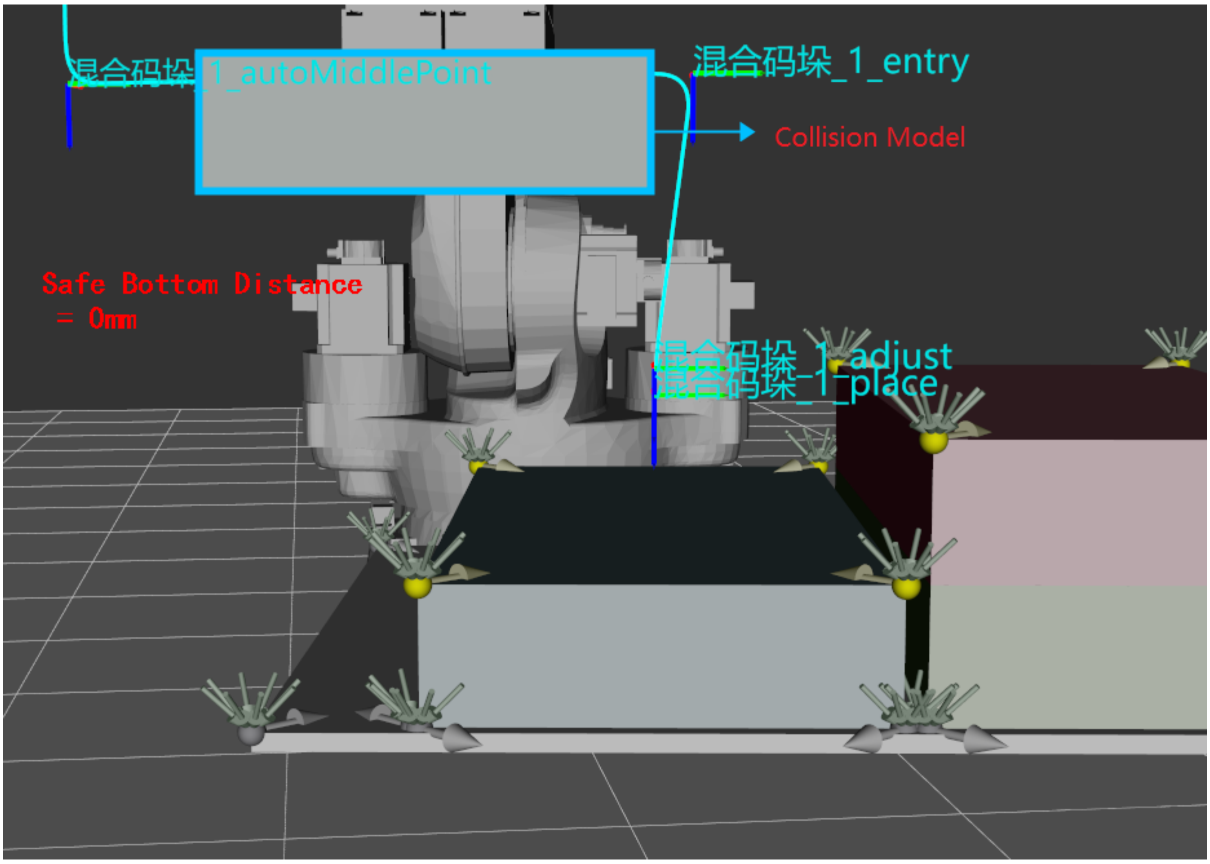
Safety Bottom Distance of cuboid: 0 mm

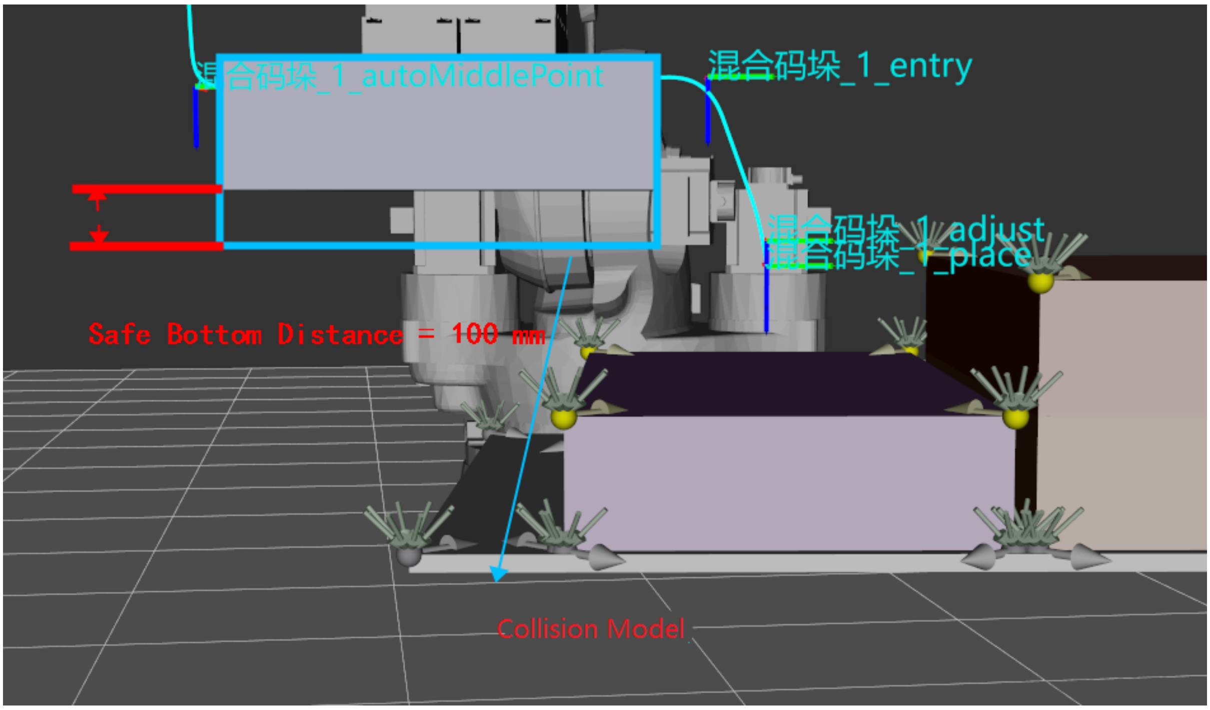
Detect Collision with Point Cloud

Detect collision with point cloud

Collision volume threshold: 50000 mm³

There is also a function “Use a safety bottom distance for the cuboid during the holding and placement in mixed-pallet”. The main purpose is to thicken the bottom of the picked box colliding model, which is used to select a safer trajectory to prevent the box from colliding with placed boxes when moving, as shown in the figure below.

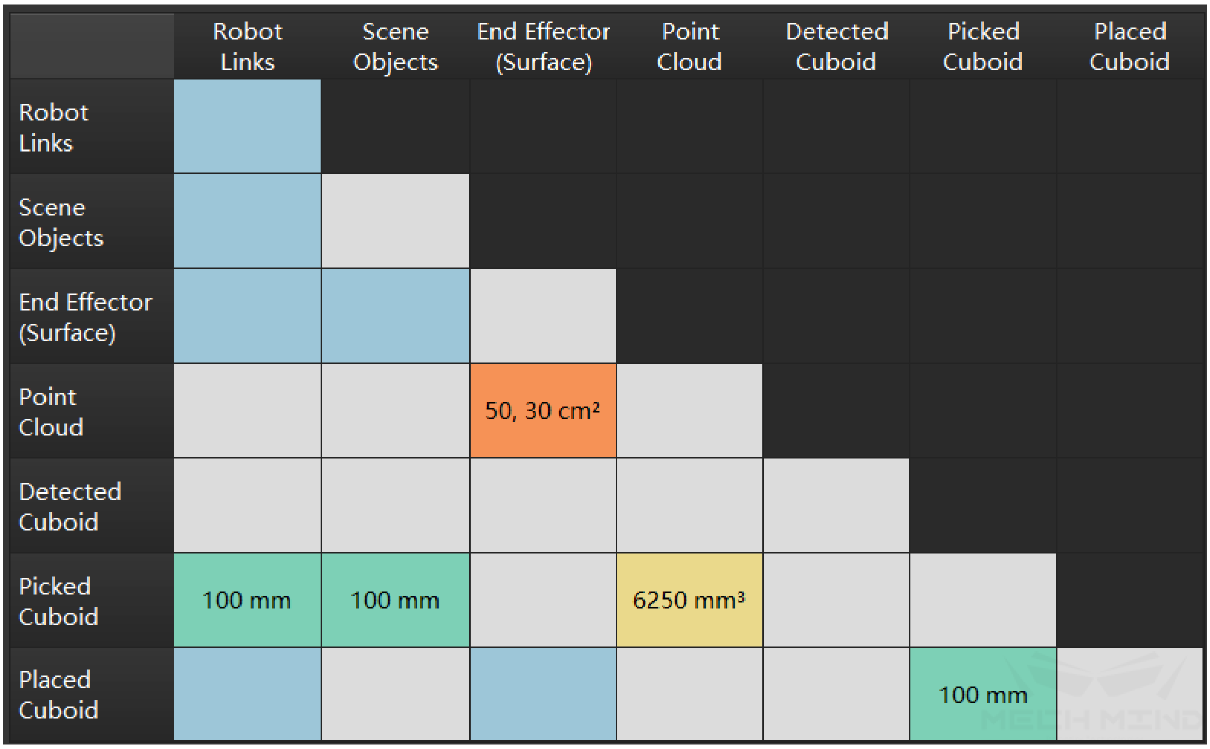




This function is currently only used for mixed palletizing and it must be turned on in mixed palletizing projects. The setting is simple. Turn on the function through

ON Use a Safety Bottom Distance for the cuboid during the holding and placement in Mixed-Pallet, and then set the safe bottom distance (recommended value is 80mm-120mm).

After the setting, the result is a collision detection scene containing the picked cuboid. The collision detection performed at this time is scene collision, self collision, point cloud collision and picked cuboid collision, as shown in the following figure.

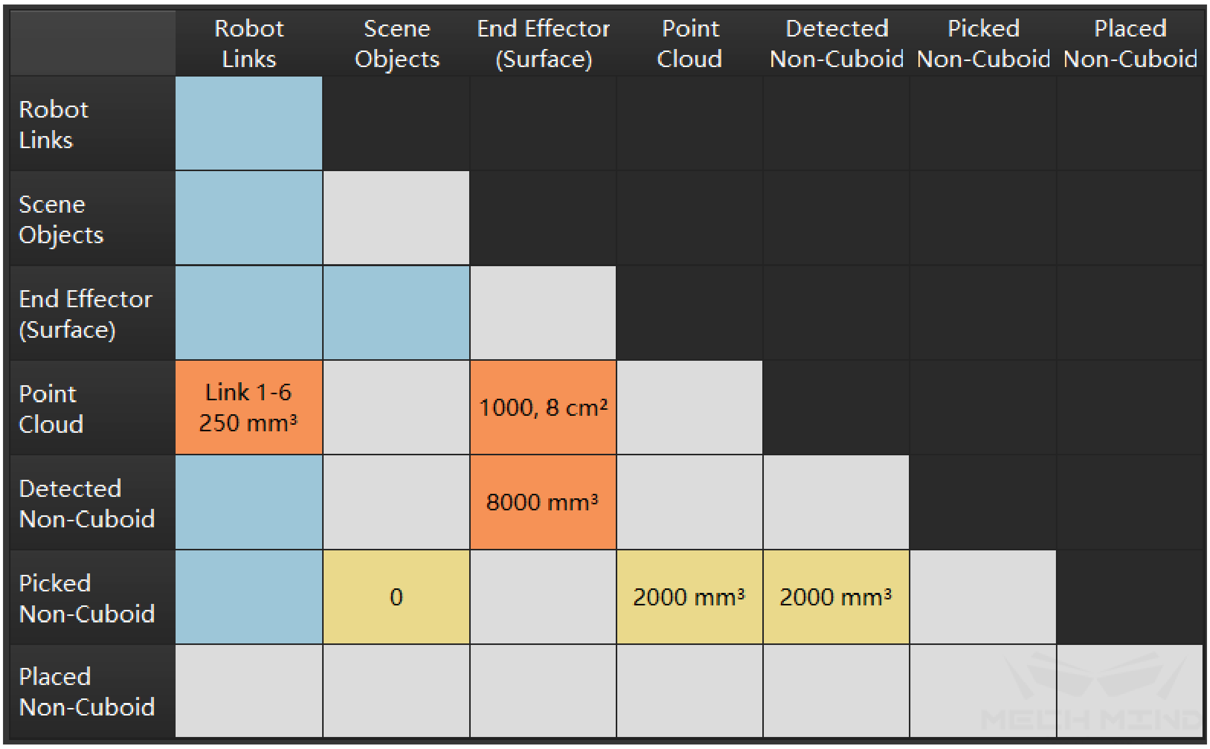


	Robot Links	Scene Objects	End Effector (Surface)	Point Cloud	Detected Cuboid	Picked Cuboid	Placed Cuboid
Robot Links	🟦						
Scene Objects	🟦	⬜					
End Effector (Surface)	🟦	🟦	⬜				
Point Cloud	⬜	⬜	50, 30 cm^2	⬜			
Detected Cuboid	⬜	⬜	⬜	⬜	⬜		
Picked Cuboid	100 mm	100 mm	⬜	6250 mm^3	⬜	⬜	
Placed Cuboid	🟦	⬜	🟦	⬜	⬜	100 mm	⬜

- **Non-cuboid**: It is also necessary to set *Collision volume threshold*, but this threshold is also used to detect the collision between the picked non-cuboid and the recognized objects, so the collision volume with the point cloud of the unrecognized object should be used as the threshold.

In both types of objects, the point cloud of the recognized objects needs to be removed. This function can be turned on through ON Remove the point cloud of the picked cuboid in *Point Cloud Configuration*.

After configuration, the result is a collision detection scene with picked non-cuboid. At this time, the collision detection performed is scene collision, self collision, point cloud collision, and picked non-cuboid collision, as shown in the following figure.



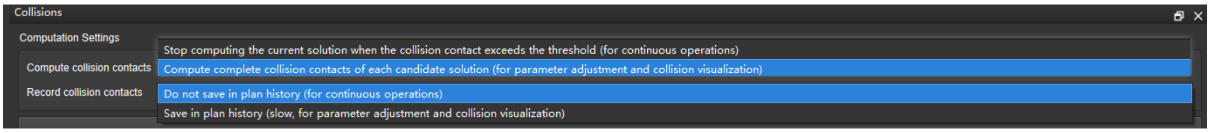
	Robot Links	Scene Objects	End Effector (Surface)	Point Cloud	Detected Non-Cuboid	Picked Non-Cuboid	Placed Non-Cuboid
Robot Links	🟦						
Scene Objects	🟦	⬜					
End Effector (Surface)	🟦	🟦	⬜				
Point Cloud	Link 1-6 250 mm^3	⬜	1000, 8 cm^2	⬜			
Detected Non-Cuboid	🟦	⬜	8000 mm^3	⬜	⬜		
Picked Non-Cuboid	🟦	0	⬜	2000 mm^3	2000 mm^3	⬜	
Placed Non-Cuboid	⬜	⬜	⬜	⬜	⬜	⬜	⬜

The Visualization of Collision Detection

There are several settings related to the visualization of collision detection.

Set Collision Detection Computing and Recording Method

The visual content of the collision result is set by selecting the different methods of *Compute collision contacts* and *Record collision contacts*, as shown in the figure below.



Collisions

Computation Settings

Compute collision contacts:
- Stop computing the current solution when the collision contact exceeds the threshold (for continuous operations)
- Compute complete collision contacts of each candidate solution (for parameter adjustment and collision visualization)

Record collision contacts:
- Do not save in plan history (for continuous operations)
- Save in plan history (slow, for parameter adjustment and collision visualization)

- *Compute collision contacts* include two computing methods: `Stop computing the current solution when the collision contact exceeds the threshold (for continuous operation)` and `Compute complete collision contacts of each candidate solution (for parameter adjustment and collision visualization)` .
- *Record collision contacts* include two recording methods: `Do not save in plan history (for continuous operations)` and `Save in plan history (slow, for parameter adjustment and collision visualization)` .

There are four differents situations through the combination of these 4 methods :

		Compute collision contacts	
		Compute complete collision contacts of each candidate solution (for parameter adjustment and collision visualization)	Stop computing the current solution when the collision contact exceeds the threshold (for continuous operation)
Record collision contacts	Save in plan history (slow, for parameter adjustment and collision visualization)	Generally used for debugging, calculate all the complete solutions for visualization, but the computing speed is slow.	The calculation stop when the collision contact exceeds the threshold, the visualization is still available, but only partial collision are displayed.In the scene with 0 tolerance for collisions, only one point of visualization is available and it is difficult to find.
	Do not save in plan history (for continuous operations)	The complete collision contact can be displayed in the plan history but the specific collision location cannot be visualized	Generally used for stable production, the collision contact does not record in the plan history, the computing speed is fast.

Normally we use the method for continuous operation at same time for computing and recording or use the method for paramter adjustment and collision visualization at the same time.

Show Collision while Planning

You need to check *Display* → *Show Collision while Planning* in the menu bar, and you can set the display duration and the expired time of the collision in the plan history.

Show Collision in Plan History

Click the relevant content in plan history to display the current pose of the robot. For more information about the plan history, please refer to *Plan History* .

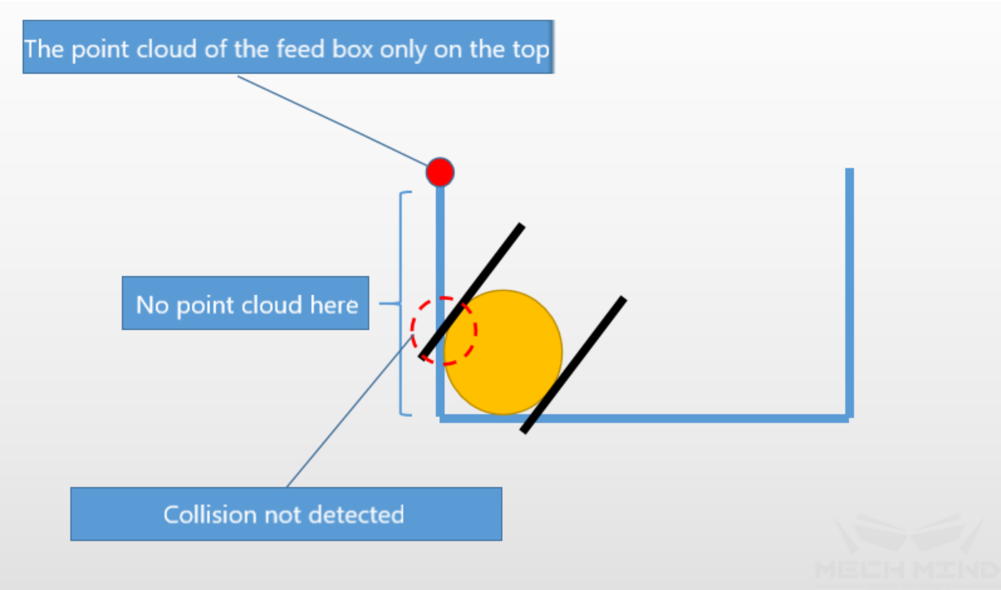
Notice

> **Attention:** Only relying on point cloud collision detection to avoid collisions with fixed objects is not reliable enough. For fixed objects, their models should be added directly to the scene and more collision constraints should be used.

This is because the set threshold of the number of collision points with the point cloud, the threshold of the collision area and volume with the point cloud are generally used to tolerate errors, such as spatial noise, recognition deviation, small deviations of jig model, etc. And if the recognized point cloud is sparse and noisy, or a large number of point clouds are missing due to reflection, the effect of point cloud collision detection may be bad.

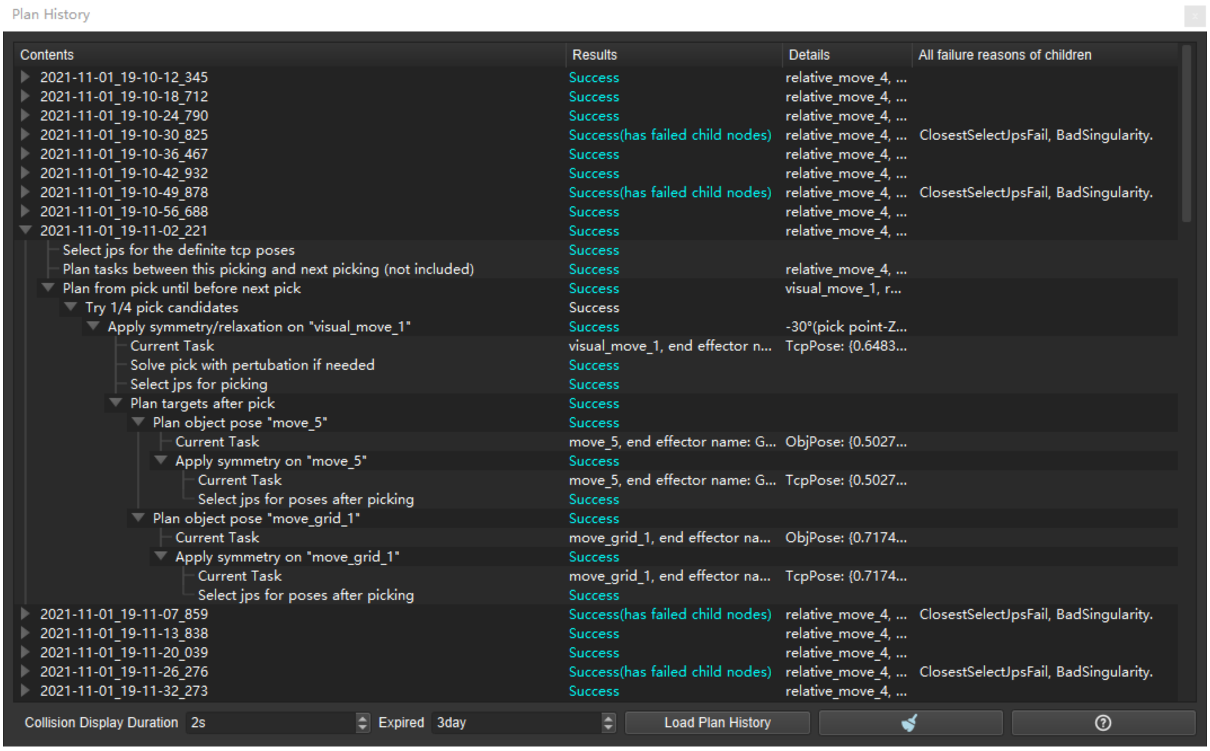
For example, in the following cases, it is not feasible to rely solely on point cloud collision detection:

1. Take the application of the crankshaft as an example, the threshold is 100 points, and these 100 points are usually used to tolerate the point cloud error of the crankshaft. If the crankshaft at the border of the feed box does not collide with the gripper jaws, and the gripper and the feed box coincide 98 points, under this situation, because the feed box does not have a scene model added, the software cannot calculate the collision and the number of point cloud collisions is smaller than the threshold. The softerware will ignore that there is collision and plan to grab, causing the jaws to collide with the feedbox.
2. In most scenarios, the point cloud is just a shell on the top surface. In case of the situation shown in the figure below, it is impossible to detect the collision between the jig and the feed box, because there is no point cloud in the feed box where the jig is located.



2.3.6 Plan History

The Plan History window expands the planning information and error description at runtime to help to debug, as shown in the following screenshot:



Contents	Results	Details	All failure reasons of children
2021-11-01_19-10-12_345	Success	relative_move_4, ...	
2021-11-01_19-10-18_712	Success	relative_move_4, ...	
2021-11-01_19-10-24_790	Success	relative_move_4, ...	
2021-11-01_19-10-30_825	Success(has failed child nodes)	relative_move_4, ...	ClosestSelectJpsFail, BadSingularity.
2021-11-01_19-10-36_467	Success	relative_move_4, ...	
2021-11-01_19-10-42_932	Success	relative_move_4, ...	
2021-11-01_19-10-49_878	Success(has failed child nodes)	relative_move_4, ...	ClosestSelectJpsFail, BadSingularity.
2021-11-01_19-10-56_688	Success	relative_move_4, ...	
2021-11-01_19-11-02_221	Success	relative_move_4, ...	
Select jps for the definite tcp poses	Success		
Plan tasks between this picking and next picking (not included)	Success	relative_move_4, ...	
Plan from pick until before next pick	Success	visual_move_1, r...	
Try 1/4 pick candidates	Success		
Apply symmetry/relaxation on "visual_move_1"	Success	-30°(pick point-Z...	
Current Task	visual_move_1, end effector n...	TcpPose: {0.6483...	
Solve pick with pertubation if needed	Success		
Select jps for picking	Success		
Plan targets after pick	Success		
Plan object pose "move_5"	Success		
Current Task	move_5, end effector name: G...	ObjPose: {0.5027...	
Apply symmetry on "move_5"	Success		
Current Task	move_5, end effector name: G...	TcpPose: {0.5027...	
Select jps for poses after picking	Success		
Plan object pose "move_grid_1"	Success		
Current Task	move_grid_1, end effector na...	ObjPose: {0.7174...	
Apply symmetry on "move_grid_1"	Success		
Current Task	move_grid_1, end effector na...	TcpPose: {0.7174...	
Select jps for poses after picking	Success		
2021-11-01_19-11-07_859	Success(has failed child nodes)	relative_move_4, ...	ClosestSelectJpsFail, BadSingularity.
2021-11-01_19-11-13_838	Success	relative_move_4, ...	
2021-11-01_19-11-20_039	Success	relative_move_4, ...	
2021-11-01_19-11-26_276	Success(has failed child nodes)	relative_move_4, ...	ClosestSelectJpsFail, BadSingularity.
2021-11-01_19-11-32_273	Success	relative_move_4, ...	

Collision Display Duration 2s Expired 3day

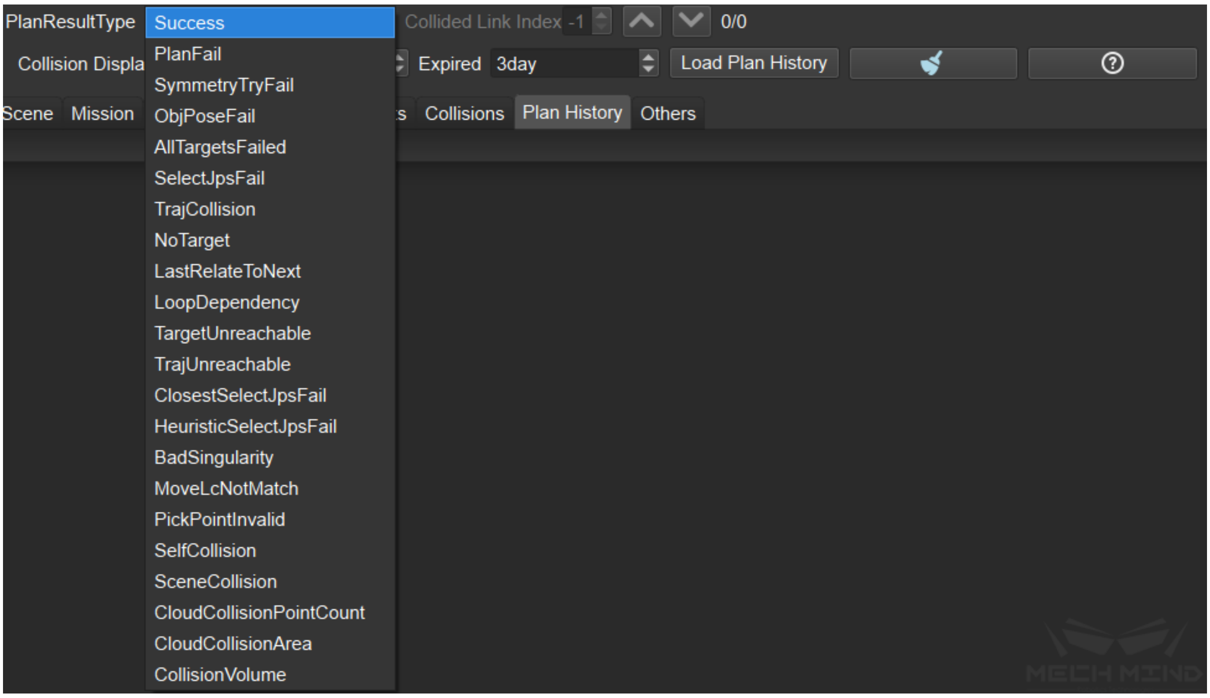
Contents: Show planning history of a specific time. After expansion, the planning content will be displayed in a hierarchy.

Graphical notices of collision results in planning history:

- Scene collision
  - Click the name of the task where the collision occurred to highlight all colliding objects.
  - Click on the colliding object to highlight itself.
- Point cloud collision
  - Click the task where the collision occurred to highlight all colliding objects.
  - Click on the volume of collision, the point cloud where the collision occurred will be highlighted.
  - If no contact point is recorded for the point cloud collision, only another object in the collision (robot, end effector, or the picked object) will be highlighted.
- Collision of an object to pick
  - When the collision occurred on an object to pick, the notice of collision is shown on the object.

Results: Show whether the plan is successful or not. If it fails, it is displayed in red. The item of failure under Contents can be expanded to view the details.

Select an item under Contents, press Ctrl+F to view the possible plan result types, as shown in the screenshot below.


PlanResultType Success

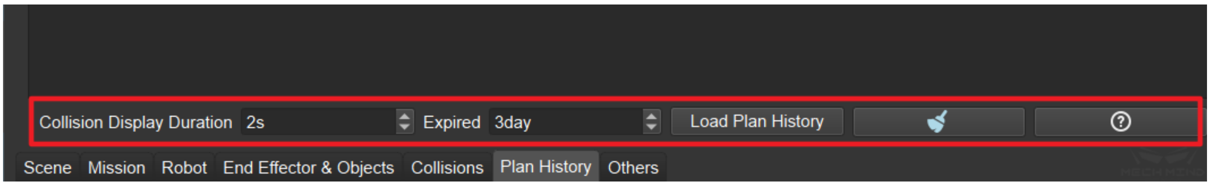
Collided Link Index -1

0/0

Collision Displa

Expired 3day

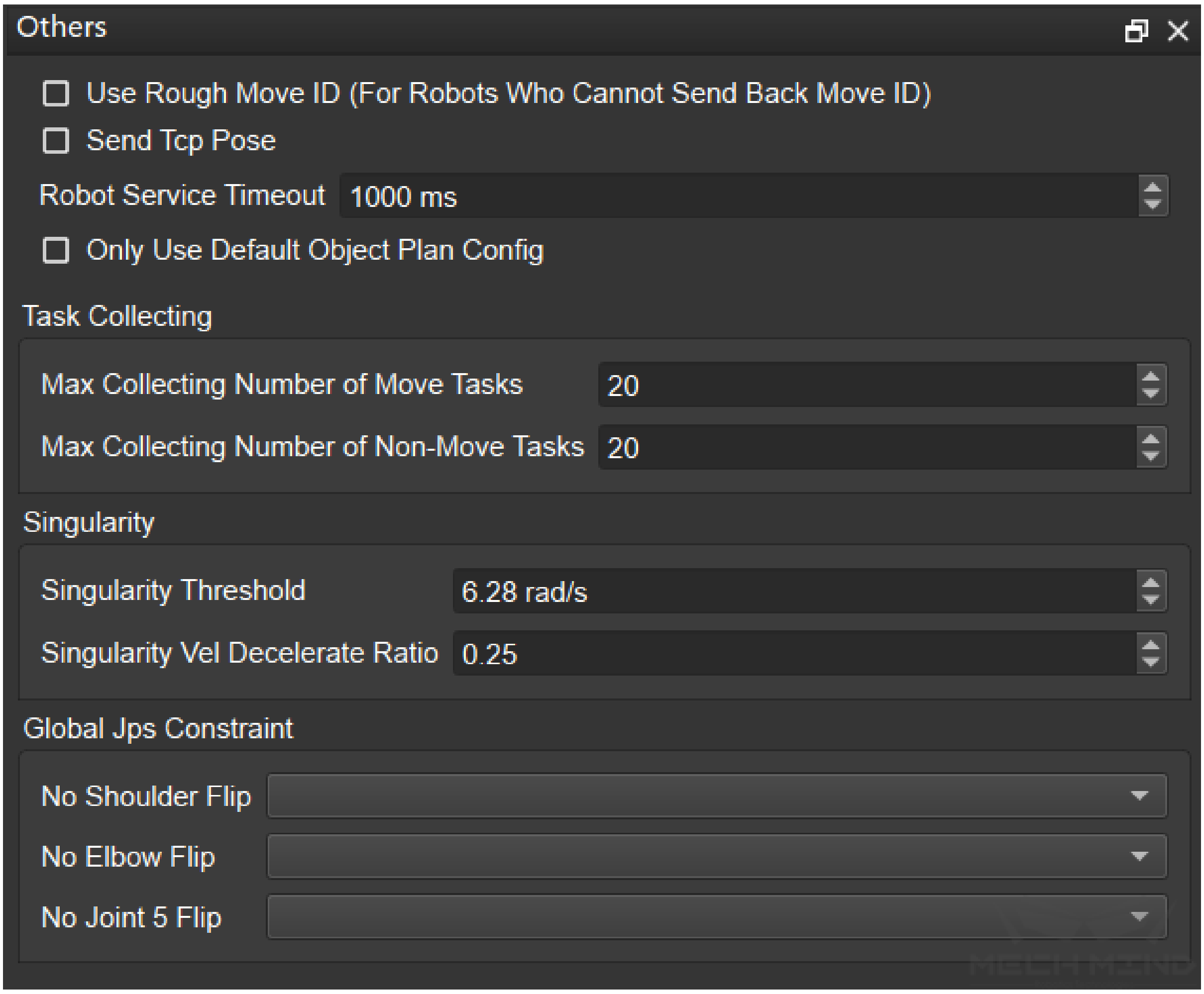
At the bottom of the window, the collision display duration, the time to expiration can be set, the plan history can be loaded, and the content of planning can be cleared, as shown in the screenshot below.


Collision Display Duration 2s

Expired 3day

2.3.7 Others

Operation Settings



Others

□ Use Rough Move ID (For Robots Who Cannot Send Back Move ID)

□ Send Tcp Pose

Robot Service Timeout: 1000 ms

□ Only Use Default Object Plan Config

Task Collecting

Max Collecting Number of Move Tasks: 20

Max Collecting Number of Non-Move Tasks: 20

Singularity

Singularity Threshold: 6.28 rad/s

Singularity Vel Decelerate Ratio: 0.25

Global Jps Constraint

No Shoulder Flip:

No Elbow Flip:

No Joint 5 Flip:

Use Rough Move ID（For Robots Who Cannot Send Back Move ID）: Calculate the rough move Id for the robot that cannot send move Id. The function is to know as accurately as possible whether the sent move is completed, so as to make the move transmission as continuous as possible, so that the robot movement is smoother. It is turned on by default, but currently this function is not stable. There may be problems in actual use. You can turn off this option when problems occur.

Send TCP Pose: Switch whether the pose sent by Mech-Viz is JPS or TCP Pose.

Robot Service Timeout : The robot service timeout period. Except for moving, all other timeout periods for communicating with the robot, such as *set_do* , *check_di* etc.

Only Use Default Object Plan Config: After it is turned on, even if there are multiple object lables, the symmetry of the default object is still used.

Task Collecting

Max Collecting Number of Move Tasks & Non-Move Tasks: On the one hand, it prevents Mech-Viz from sending too many points after falling into an infinite loop, and on the other hand, it limits the number of commands Mech-Viz sends each time.

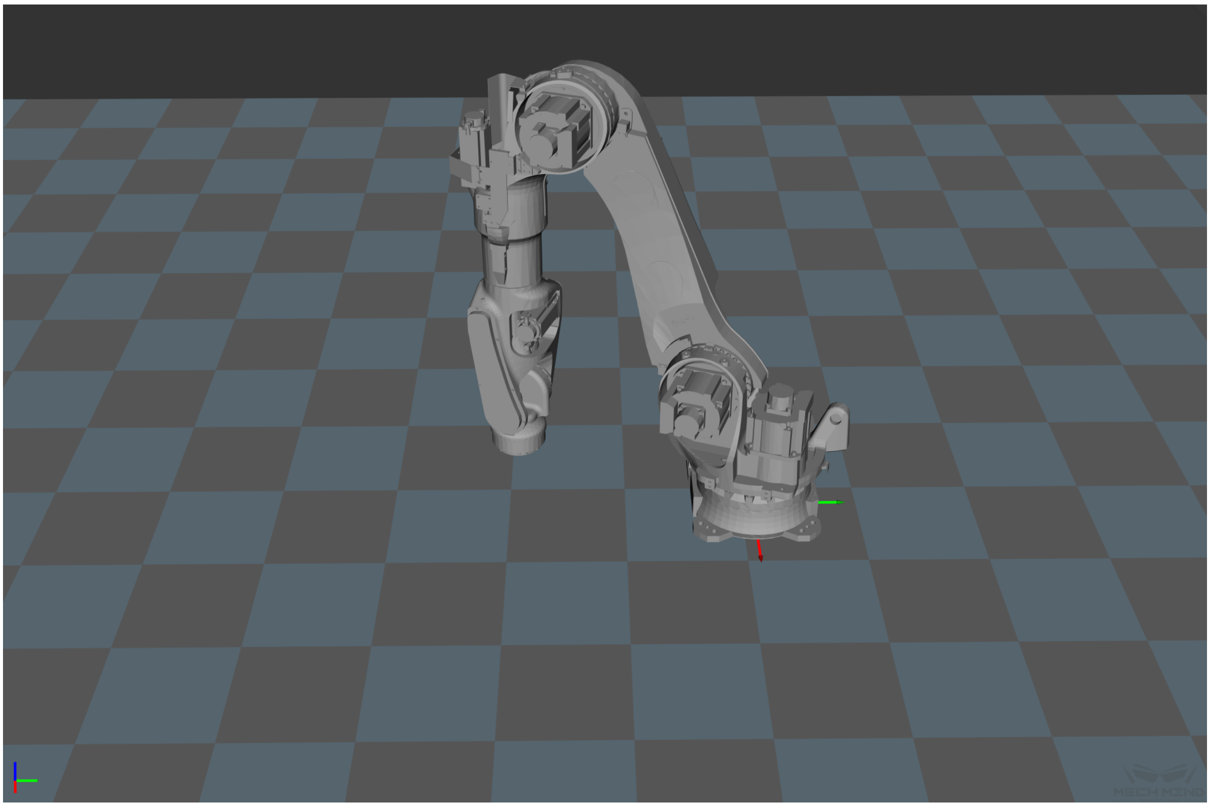
Singularity

Singularity Threshold: During path planning, it determines whether the threshold of the singular point is reached (maximum angular velocity of joints in radian system), generally the default is sufficient. If some robots are more sensitive to singularity detection (shown as: the software simulation path is normal, but the real robot follows the same path but reports the singularity error), the value can be appropriately reduced.

Singularity Vel Decelerate Ratio : When a singularity occurs and the speed reduction ratio is not lower than this value, it is considered that the singularity can be solved by speed reduction. Reducing this value can reduce the probability of software planning failure due to “severe singularity” .

Global Jps Constraint

No Shoulder Flip/NO Elbow Flip/No Joint 5 Flip : You can choose `when holding an object` or `the whole execution process` to reduce unnecessary turning of the robot. However, setting the shoulder/elbow/Joint 5 to “not turn” is not necessarily the best choice. Examples are as follows:

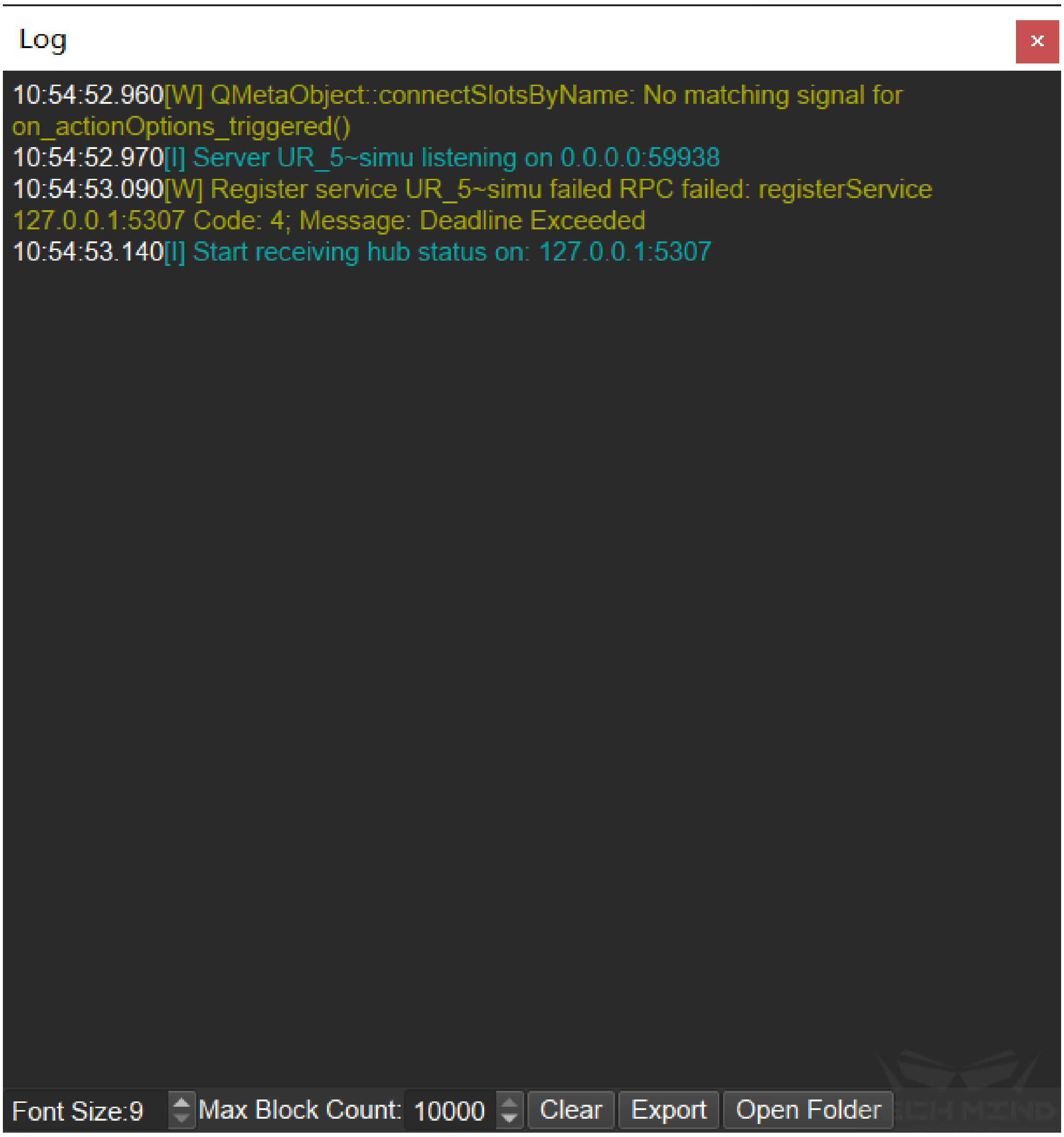


If there is the above posture during the movement, it is known that the robot Joint 5 flip/shoulder flip/elbow flip is the relationship between the axes. If the wrist flip is forcibly specified, it may cause the robot to turn more. Therefore, you should only specify `No Shoulder Flip` and `No Elbow Flip`.

Note: The priority of *JPs Constraints* for movement tasks is higher than the global settings here. If there is no setting in movement tasks, the global settings will take effect.

2.3.8 Log

The interface as shown in *Figure 1* displays the status information of Mech-Viz after startup.



```
10:54:52.960[W] QMetaObject::connectSlotsByName: No matching signal for
on_actionOptions_triggered()
10:54:52.970[I] Server UR_5~simu listening on 0.0.0.0:59938
10:54:53.090[W] Register service UR_5~simu failed RPC failed: registerService
127.0.0.1:5307 Code: 4; Message: Deadline Exceeded
10:54:53.140[I] Start receiving hub status on: 127.0.0.1:5307
```

Figure 1. Log Interface

The log levels include Debug, Info, Warning, Critical, and Fatal, listed in ascending order of urgency.

The log level can be set in *Settings* → *Log Level*. When a level is selected, the console will print the logs of the selected level and above, as shown in *Figure 2*.

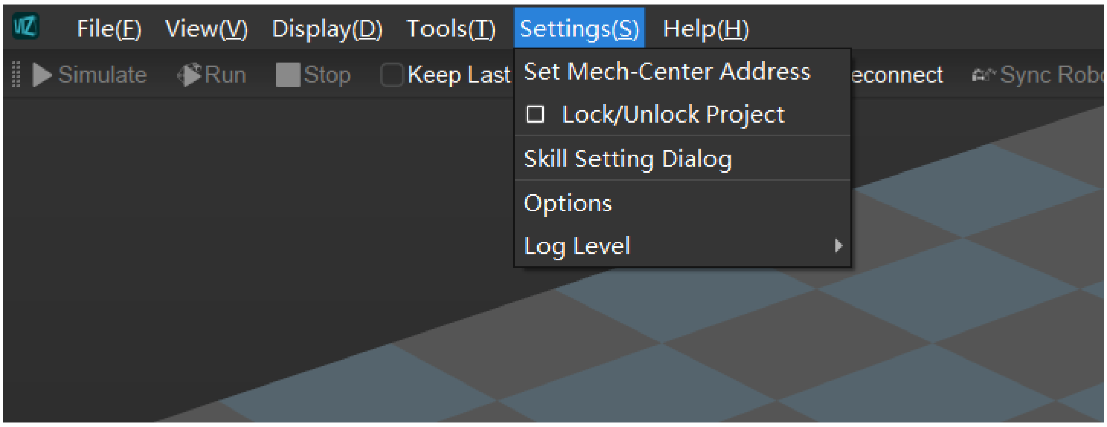
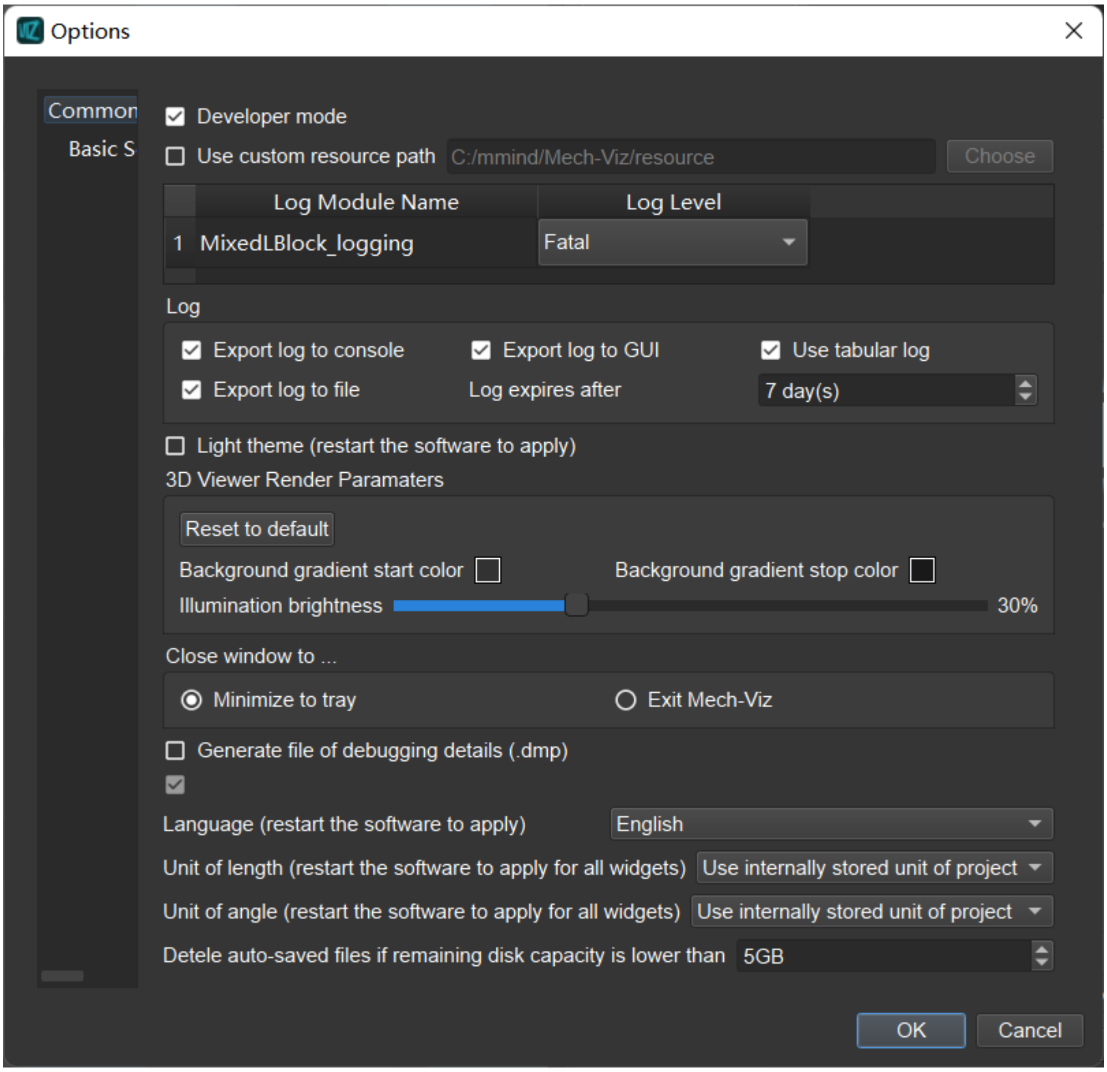


Figure 2. Log Level

Other settings of log can be found in *Settings* → *Options* → *Common Settings*, as shown in *Figure 3*.



Options

Common
Basic S

☑ Developer mode

□ Use custom resource path C:/mmind/Mech-Viz/resource Choose

	Log Module Name	Log Level
1	MixedLBlock_logging	Fatal

Log

☑ Export log to console ☑ Export log to GUI ☑ Use tabular log

☑ Export log to file Log expires after 7 day(s)

□ Light theme (restart the software to apply)

3D Viewer Render Paramaters

Reset to default

Background gradient start color □ Background gradient stop color ■

Illumination brightness 30%

Close window to ...

● Minimize to tray ○ Exit Mech-Viz

□ Generate file of debugging details (.dmp)

☑

Language (restart the software to apply) English

Unit of length (restart the software to apply for all widgets) Use internally stored unit of project

Unit of angle (restart the software to apply for all widgets) Use internally stored unit of project

Detele auto-saved files if remaining disk capacity is lower than 5GB

OK Cancel

Figure 3. Log Settings

CHAPTER

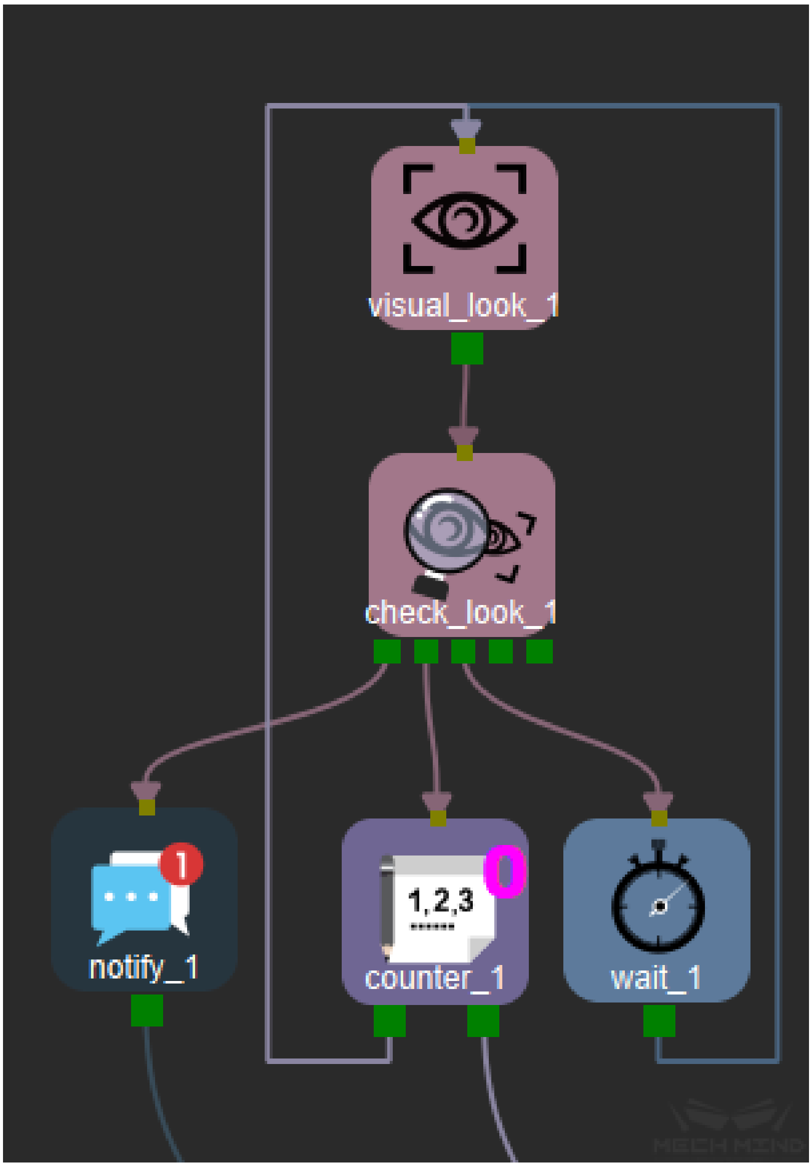
THREE

INSTRUCTIONS

3.1 Basics of Building a Project

3.1.1 Software Running Process

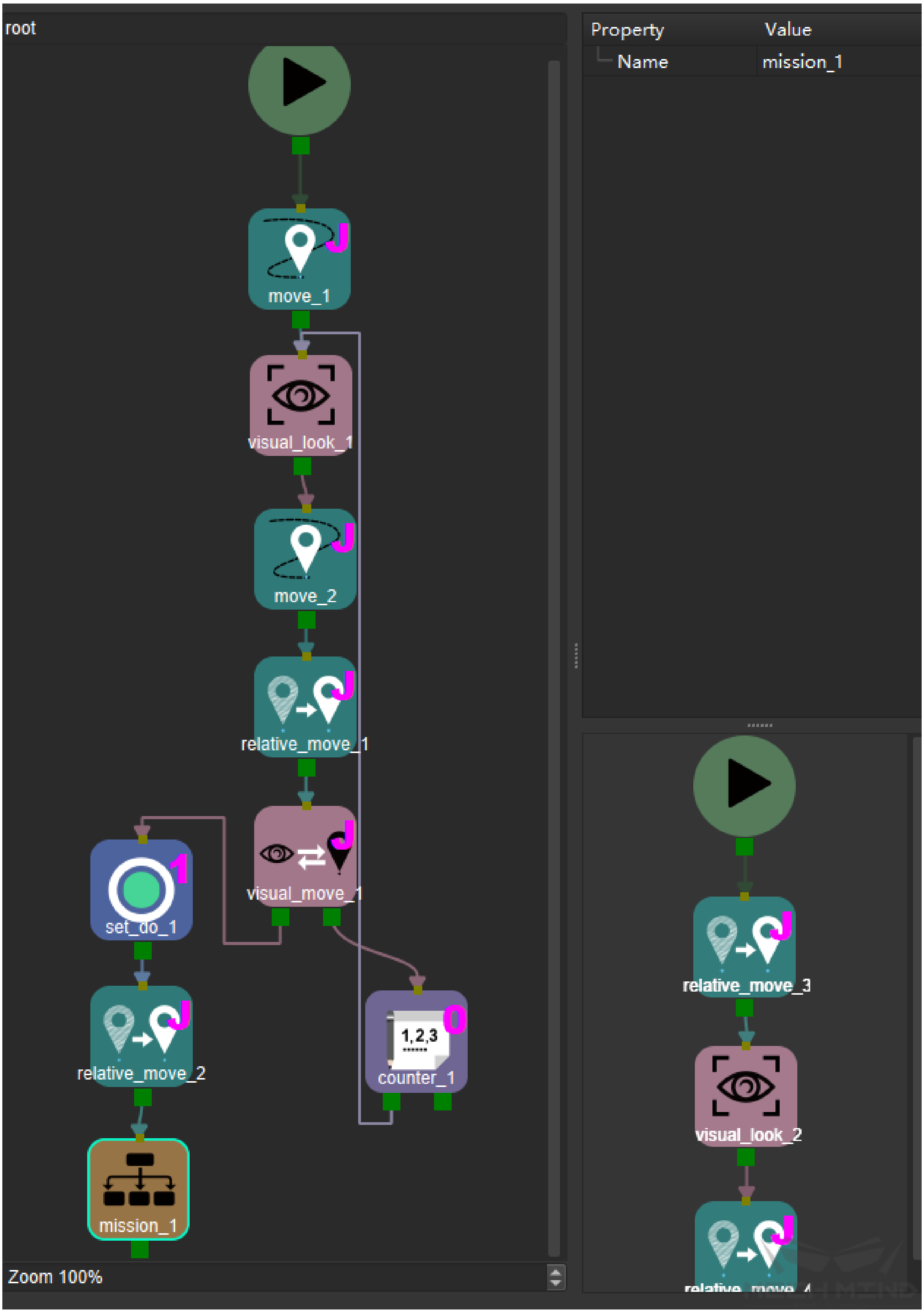
The software will serially control the operations of robot according to the sequence of the modules. For some tasks outputting different results, you can connect different tasks to different output ports of the module for branch processing. A loop of a process can be realized through connecting the output port of a module with the input of a another module in front of it, as shown in the example



After running the *visual_look* module, a variety of visual results will appear. If there is a result, you can use it to pick. When there is no result, you can run the *visual_look* again to take a picture and check the result. If the visual processing is not completed, you can set the waiting time to check the visual results again later. Multiple output ports are used to connect different tasks to complete all process logic for all possibilities.

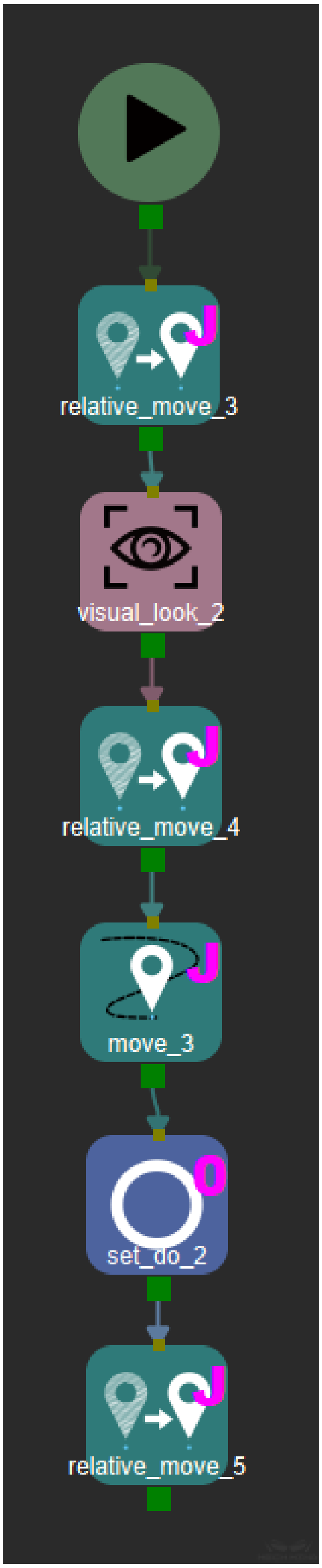
3.1.2 Multi-Level Mission Processing

If the program involves multiple tasks with different functions, you can use the “mission” module to manage multi-level task.As shown in the figure, the tasks of the same type are included in a “mission” module. Through its name and thumbnail on the bottom right you can understand the function of this “mission” intuitively and it makes modification as well as maintenance of function easier.



Property	Value
Name	mission_1

You can double-click the “mission” module to enter the sub-interface of this mission. As shown in the figure, each “mission” is equivalent to a subroutine, the execution logic is the same as in the main interface and using right-click + “Navigate Up” to return to the main interface to continue editing. When the process reaches the “mission” module, it starts to execute the “mission” process. After executing the last module in the “mission”, it jumps out and continues to execute the modules in main interface.

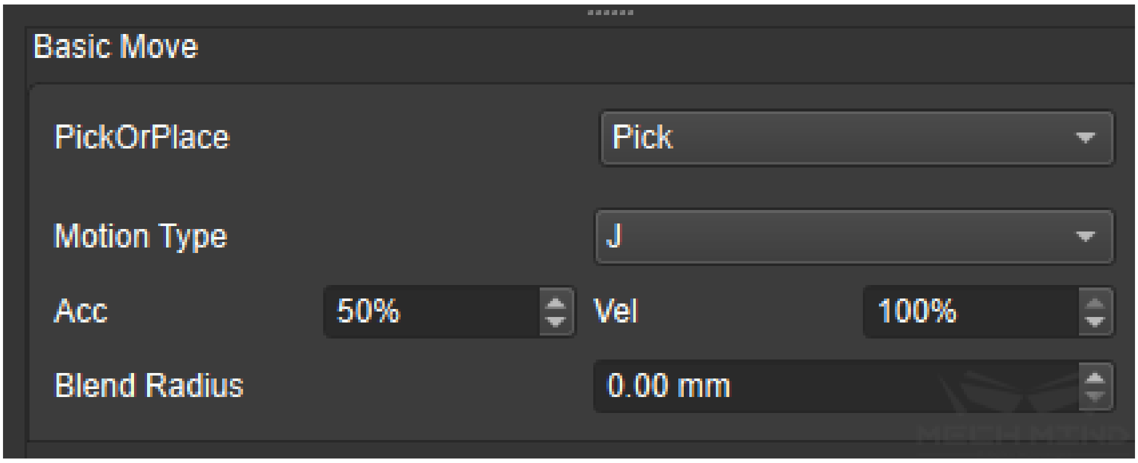

relative_move_3
visual_look_2
relative_move_4
move_3
set_do_2
relative_move_5

3.1.3 Task Type

When the software controls the movement of the robot, it will plan the trajectory according to movement tasks as well as non-movement tasks in advance and send motion commands to the robot to ensure the continuity of robot motion.

Movement Tasks

As shown in figure, the mobile task is used to control the robot to move according to the planned trajectory, motion type, speed, acceleration and other parameters in the software;



Basic Move

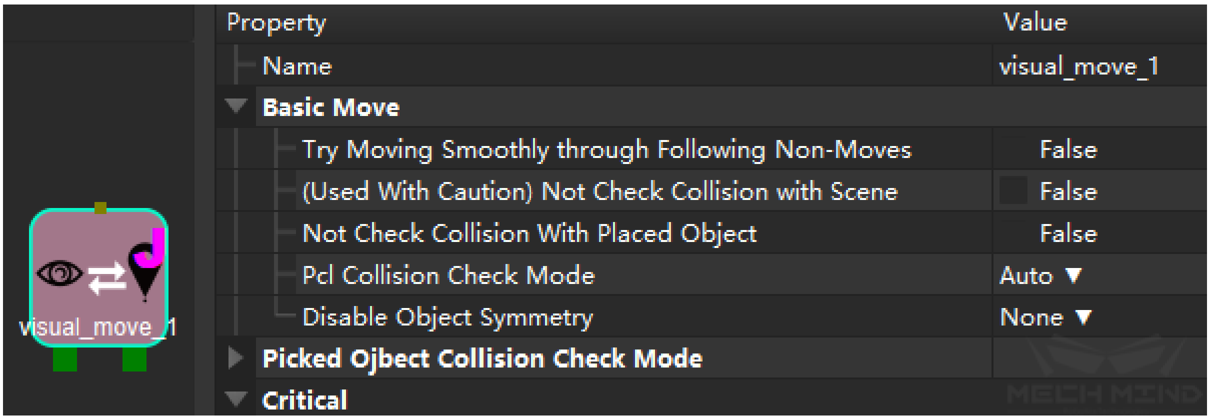
PickOrPlace: Pick

Motion Type: J

Acc: 50% Vel: 100%

Blend Radius: 0.00 mm

in addition to setting *Pick-Hold-Place* to pick or place object, you can also set whether or not to check for point cloud collisions, turn off object symmetry, make the software plan the robot’ s optimal path and pick-place-pose, as shown in figure



Property	Value
Name	visual_move_1
Basic Move	
Try Moving Smoothly through Following Non-Moves	False
(Used With Caution) Not Check Collision with Scene	False
Not Check Collision With Placed Object	False
Pcl Collision Check Mode	Auto ▼
Disable Object Symmetry	None ▼
Picked Ojbect Collision Check Mode	
Critical	

Optional Checking for Point Cloud Collisions

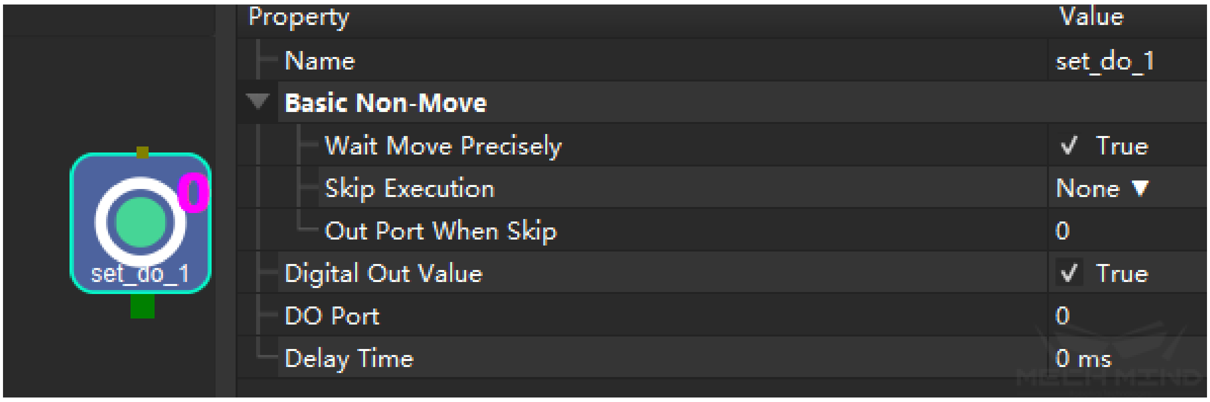
After checked the *Point Cloud Configuration* as true, point cloud collisions in vision-related movements will be checked by default , those vision-related movements include visual_move and relative_move which depends on visual_move. You can also set check or not by yourself. For example, if the picked object will definitely collide with other objects during the extraction process and this type of collision is within the allowable range, you can set this option to “notCheck” to turn checking point cloud collision off, otherwise the software will choose not to grab this object so that the grabbing efficiency will be affected;

Optional Disable for Object Symmetry

The movement tasks with `Obj Pose` as *Move Target Type* , such as the *move* module and the modules of *Pallet* that select the option `ObjPose` in Move Target type, you can set whether or not to turn off the symmetry of the object according to the needs of the scene. For example, some object has to be picked or placed in a fixed pose , you can set “disbaleObjSymm” to true to make the software planed trajectory unique.

Non-movement Tasks

Non-movement tasks include tasks such as calling for visual services, setting and receiving status of external interface, waiting, checking visual results, and other non-controlling-movement tasks of robot; according to the application, it can be set to wait for robots to move in place before execution or to execute them in advance to optimize the beat, as shown in figure



Property	Value
Name	set_do_1
Basic Non-Move	
Wait Move Precisely	✓ True
Skip Execution	None ▼
Out Port When Skip	0
Digital Out Value	✓ True
DO Port	0
Delay Time	0 ms

Summary

Both movement and non-movement tasks can be set to wait or not. If the robot needs to run continuously and reduce pauses, you can set not to wait.

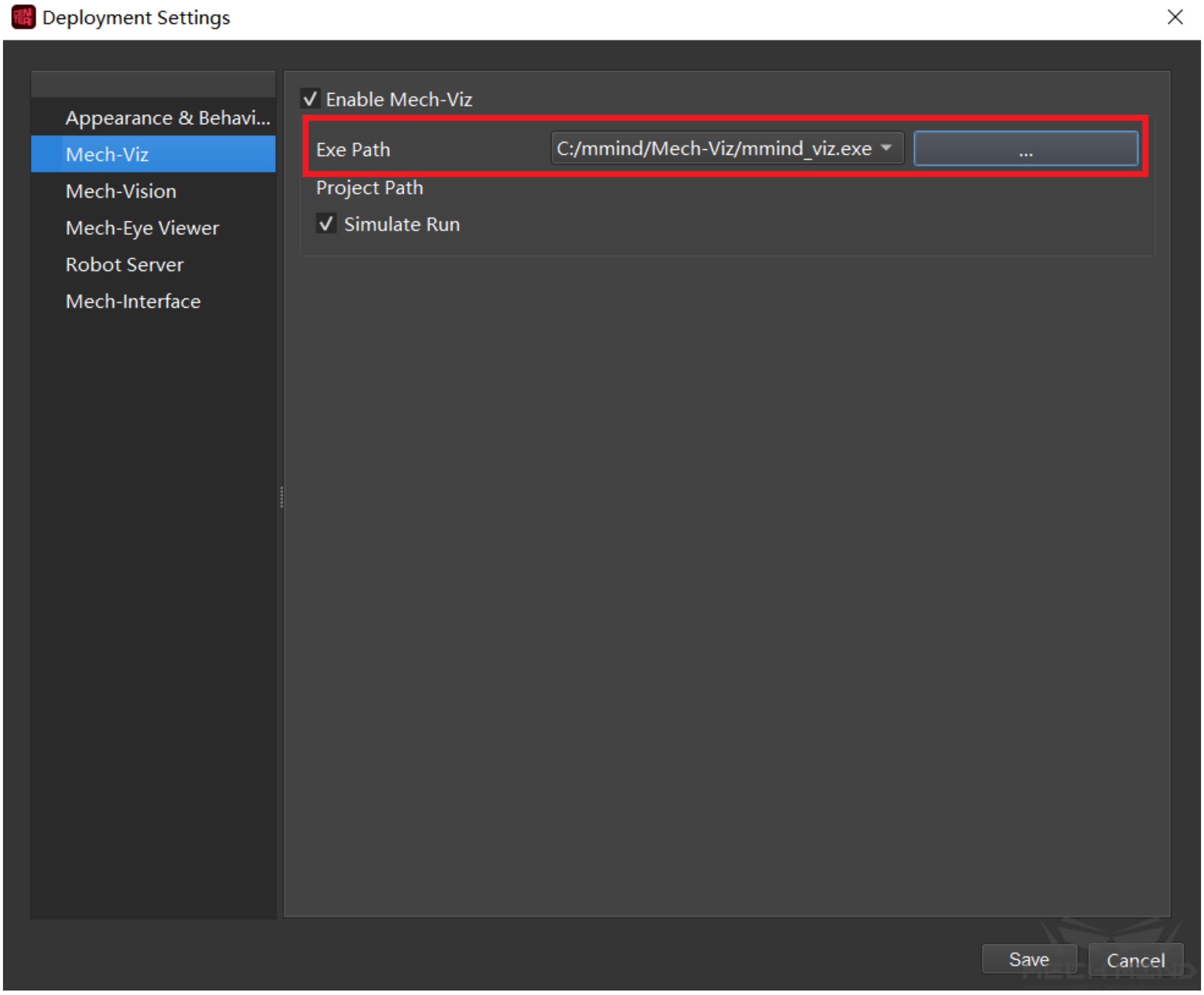
Application cases:

- Powering the sucker up in advance during the robot’ s movement can speed up the beat of picking;
- Checking the external signals in advance during the robot’ s movement, such as the signal that object is in place, can shorten the time of the movement and waiting for the signal, and optimize the overall operation cycle.

3.2 Create a Project

3.2.1 Set Mech-Viz Execution Path

Launch Mech-Center, in Deployment Settings, select Enable Mech-Viz, set the Exe Path, as shown in the screenshot below.



Deployment Settings

☑ Enable Mech-Viz

Exe Path: C:/mmind/Mech-Viz/mmind_viz.exe

Project Path

☑ Simulate Run

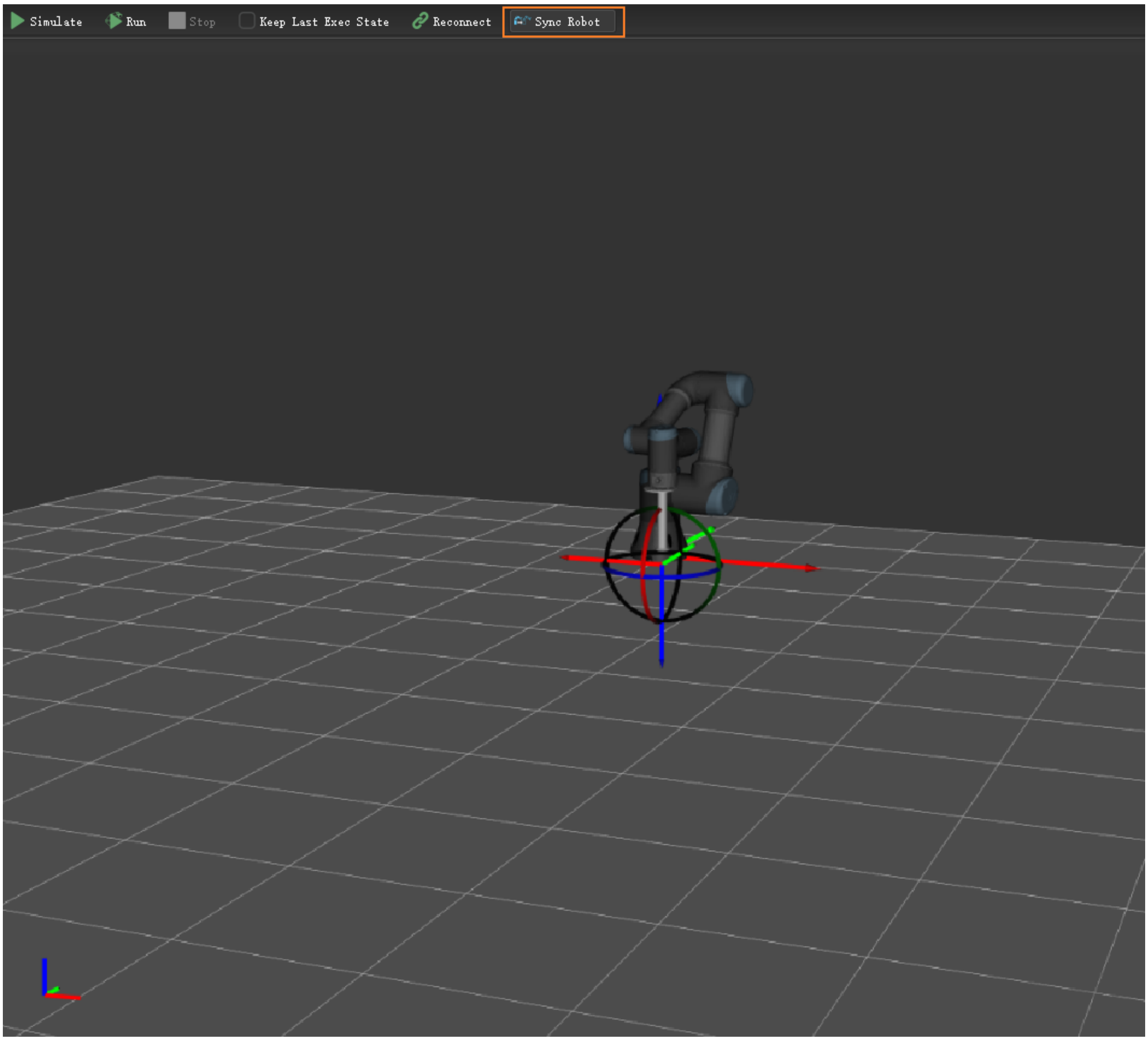
3.2.2 Establish Connection with the Real Robot

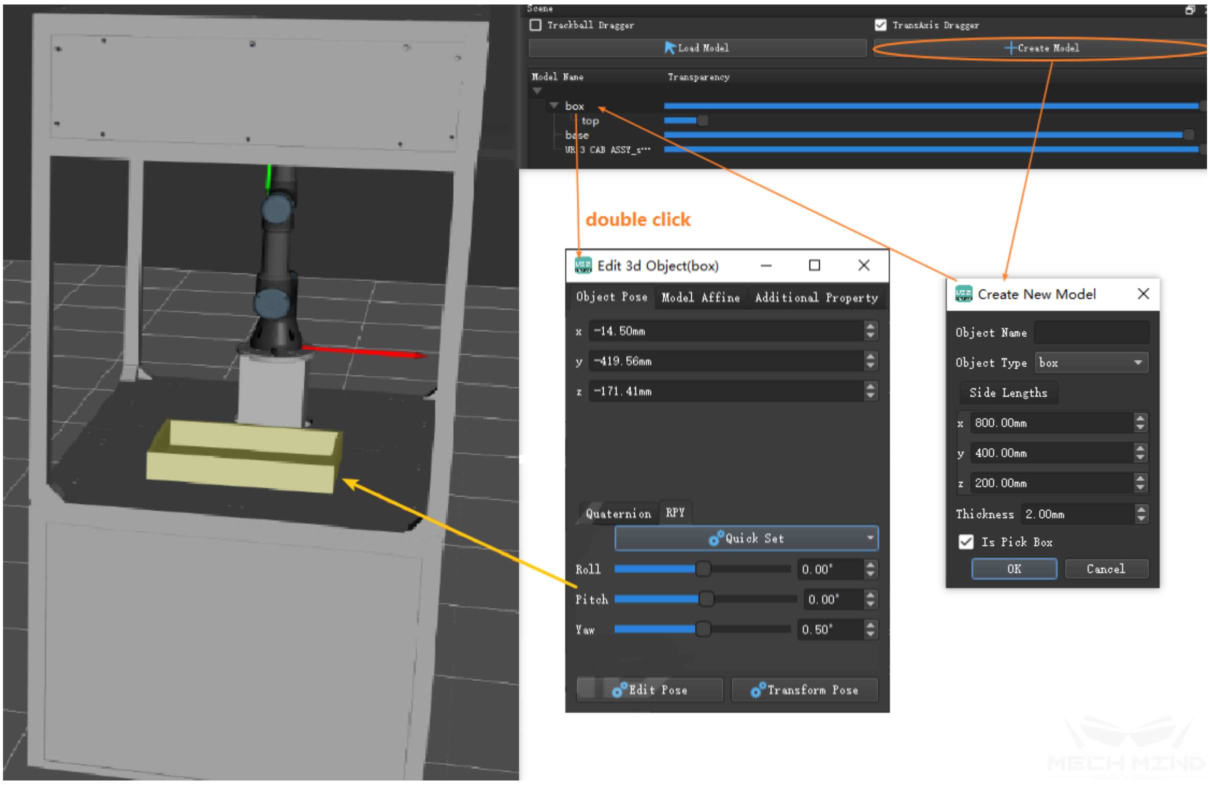
Start Mech-Viz from Mech-Center.

In Mech-Viz, select the robot model under Configure Robot in the Robot tab, save the project, and select *Autoload Current Project* in the bar on the top.

Click *Connect Robot* in Mech-Center.

Click *Sync Robot* to obtain the relative position of the real robot and the scene. At this point, the model(s) of the surrounding object(s) can be imported in the Scene tab based on the on-site circumstances. As shown in the screenshot below.


Simulate
Run
Stop
Keep Last Exec State
Reconnect
Sync Robot



x −14.50mm
y −419.56mm
z −171.41mm

Roll 0.00°
Pitch 0.00°
Yaw 0.50°

Object Name
Object Type box
Side Lengths
x 800.00mm
y 400.00mm
z 200.00mm
Thickness 2.00mm
☑ Is Pick Box

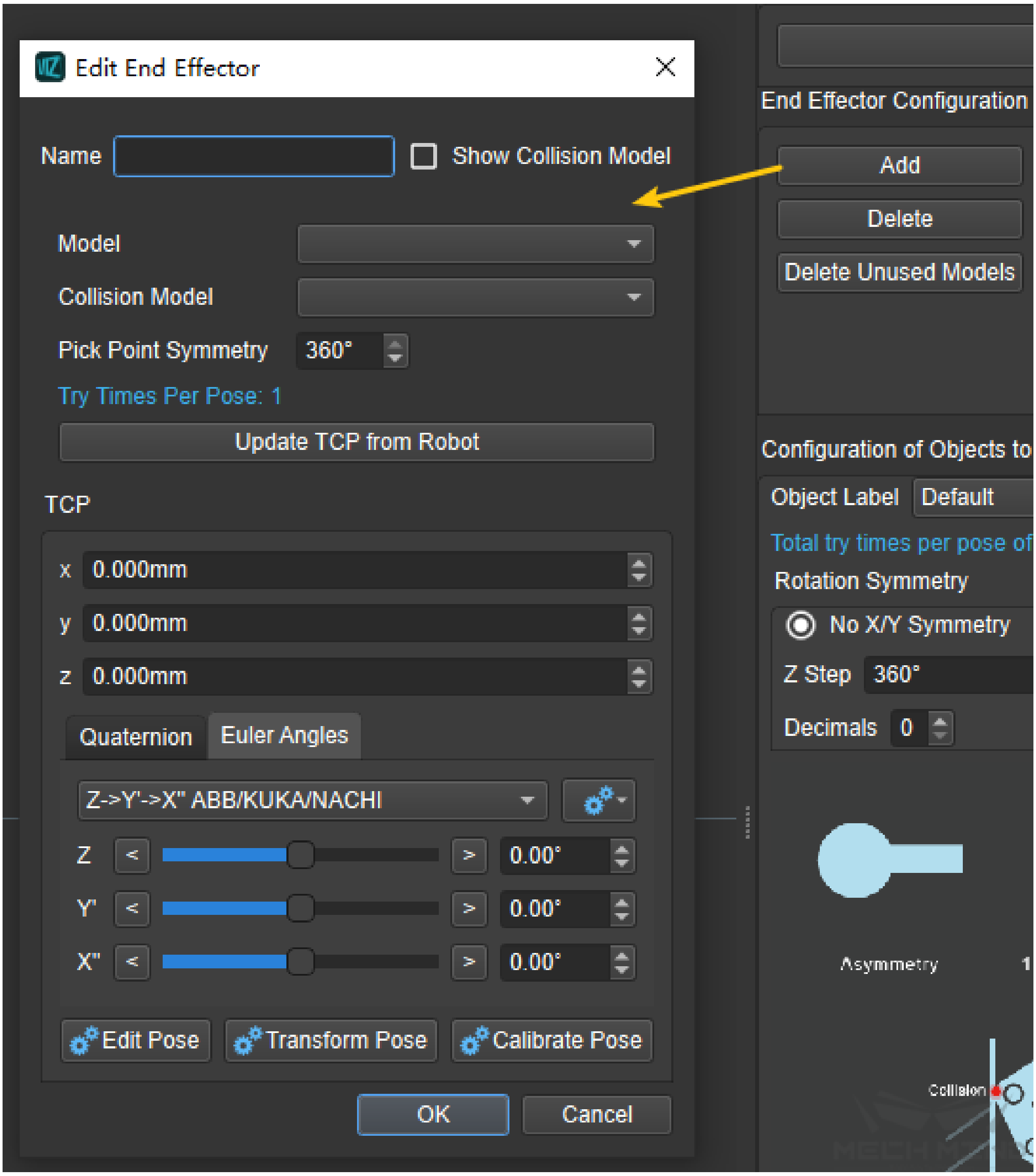
After determining the type and size of an object model, double-click the model name or directly double-click the model in the 3D simulation area to pop up the editing interface to adjust the pose of the model.

3.2.3 Writing a Robot Program

Taking a vision-guided pick-and-place mission as an example, the object to pick is a rectangular parallelepiped, with 180° symmetry around the z-axis, and the end effector is a circular suction cup with 0° symmetry at the pick point.

Set Symmetricity

Click *Add* under **End Effector Configuration** in the **End Effector & Objects** tab, set **Pick Point Symmetry** to 0°; in the same tab, under **Configuration of Objects to Pick (For move planning)**, set the rotational symmetry around the Z-axis (**Z Step** under **Rotation Symmetry**) of the target object to 180°.


Edit End Effector

Name

□ Show Collision Model

Model

Collision Model

Pick Point Symmetry 360°

Try Times Per Pose: 1

TCP

x 0.000mm

y 0.000mm

z 0.000mm

Quaternion | Euler Angles

Z->Y'->X" ABB/KUKA/NACHI

Z 0.00°

Y' 0.00°

X" 0.00°

End Effector Configuration

Configuration of Objects to

Object Label Default

Total try times per pose of

Rotation Symmetry

● No X/Y Symmetry

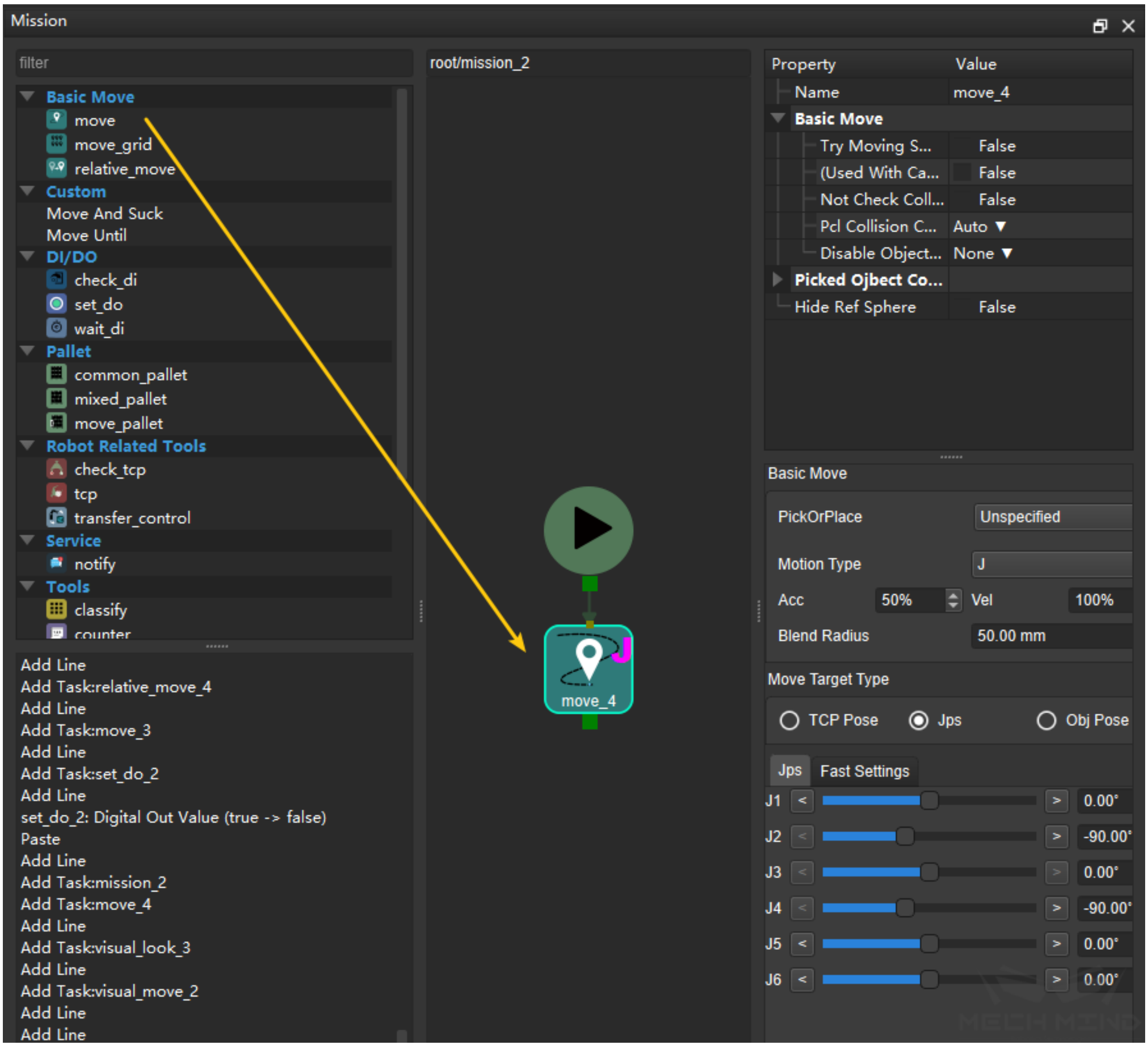
Z Step 360°

Decimals 0

Write A Simple Teach-In Program

Try to control the robot’s movement through the graphical modules.

1. As shown in the figure, pull a “move” module from the mission set interface on the left to the programming interface in the middle.



Add Line
Add Task:relative_move_4
Add Line
Add Task:move_3
Add Line
Add Task:set_do_2
Add Line
set_do_2: Digital Out Value (true -> false)
Paste
Add Line
Add Task:mission_2
Add Task:move_4
Add Line
Add Task:visual_look_3
Add Line
Add Task:visual_move_2
Add Line
Add Line

Property	Value
Name	move_4
Basic Move	
Try Moving S...	False
(Used With Ca...	False
Not Check Coll...	False
Pcl Collision C...	Auto ▼
Disable Object...	None ▼
Picked Ojbect Co...	
Hide Ref Sphere	False

Basic Move

PickOrPlace: Unspecified

Motion Type: J

Acc: 50% Vel: 100%

Blend Radius: 50.00 mm

Move Target Type

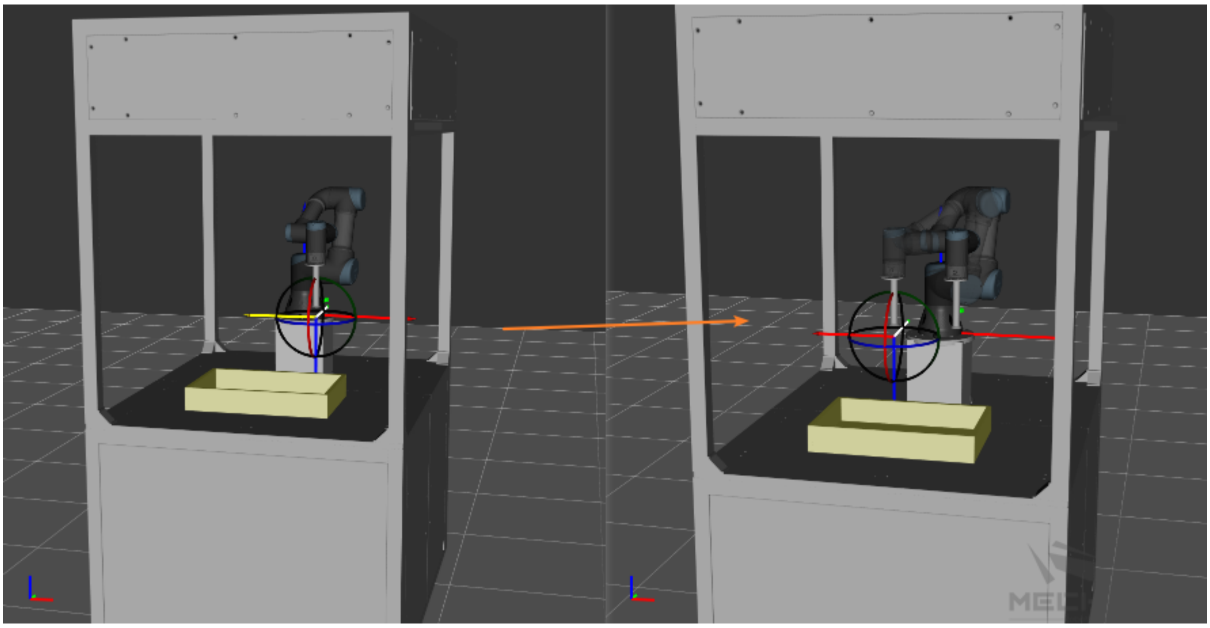
○ TCP Pose ● Jps ○ Obj Pose

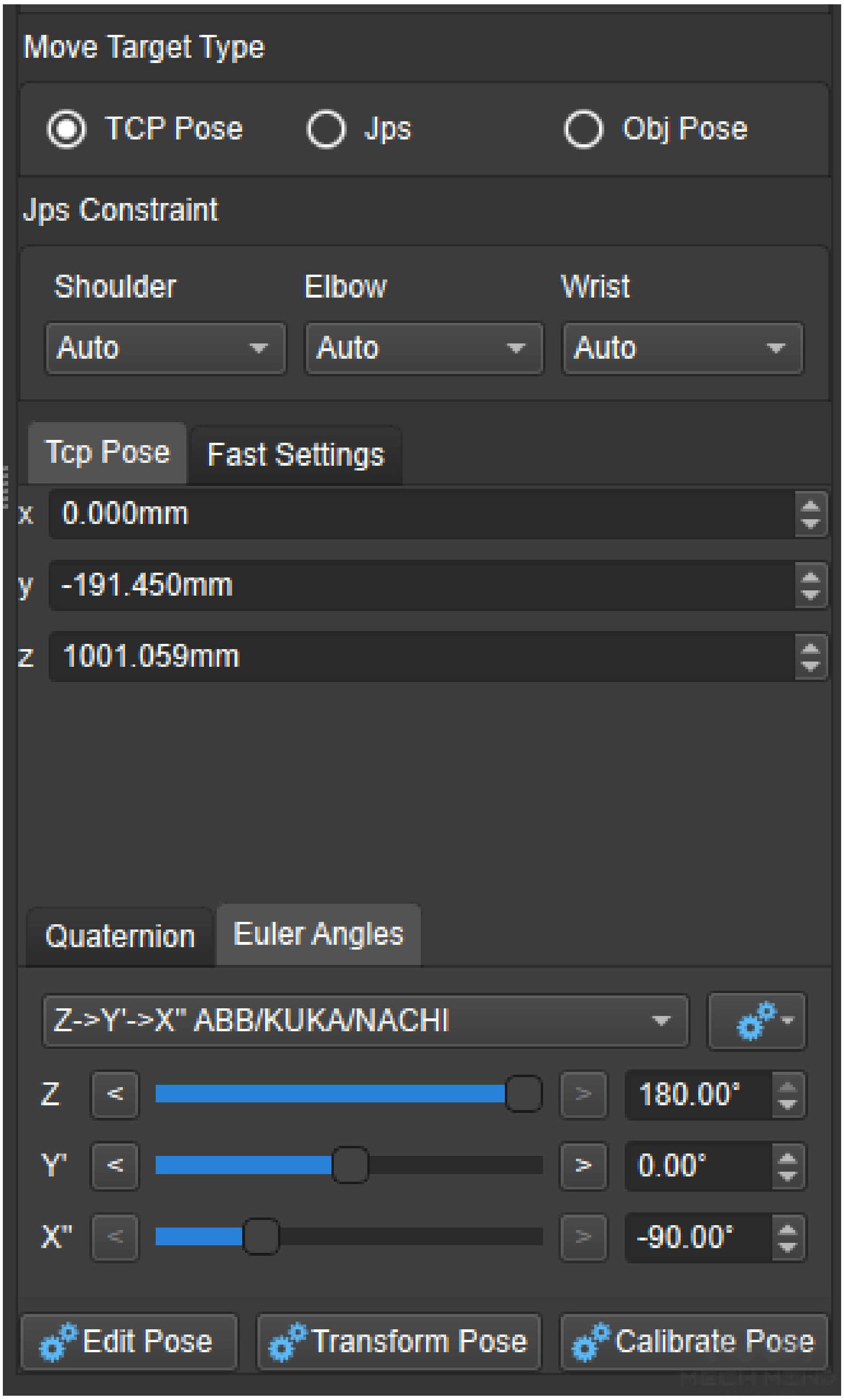
Jps | Fast Settings

J1	0.00°
J2	-90.00°
J3	0.00°
J4	-90.00°
J5	0.00°
J6	0.00°

1. There are many ways to set the pose of the target point. The target pose can be set by dragging in the 3D simulation, or through the parameters on the right.

As shown in the screenshot below.





Move Target Type

● TCP Pose ○ Jps ○ Obj Pose

Jps Constraint

Shoulder: Auto

Elbow: Auto

Wrist: Auto

Tcp Pose | Fast Settings

x 0.000mm

y -191.450mm

z 1001.059mm

Quaternion | Euler Angles

Z->Y'->X" ABB/KUKA/NACHI

Z 180.00°

Y' 0.00°

X" -90.00°

1. After connecting the input of the "move" task to the rest of the tasks, the move can be simulated by running the program, i.e., clicking Simulation in the upper left corner.

Tip: The "move" task is usually used to set the initial position, the photo-taking position, and the transition during picking and placing. Normally, to reduce the possibility of singularity, joint motion (**J**

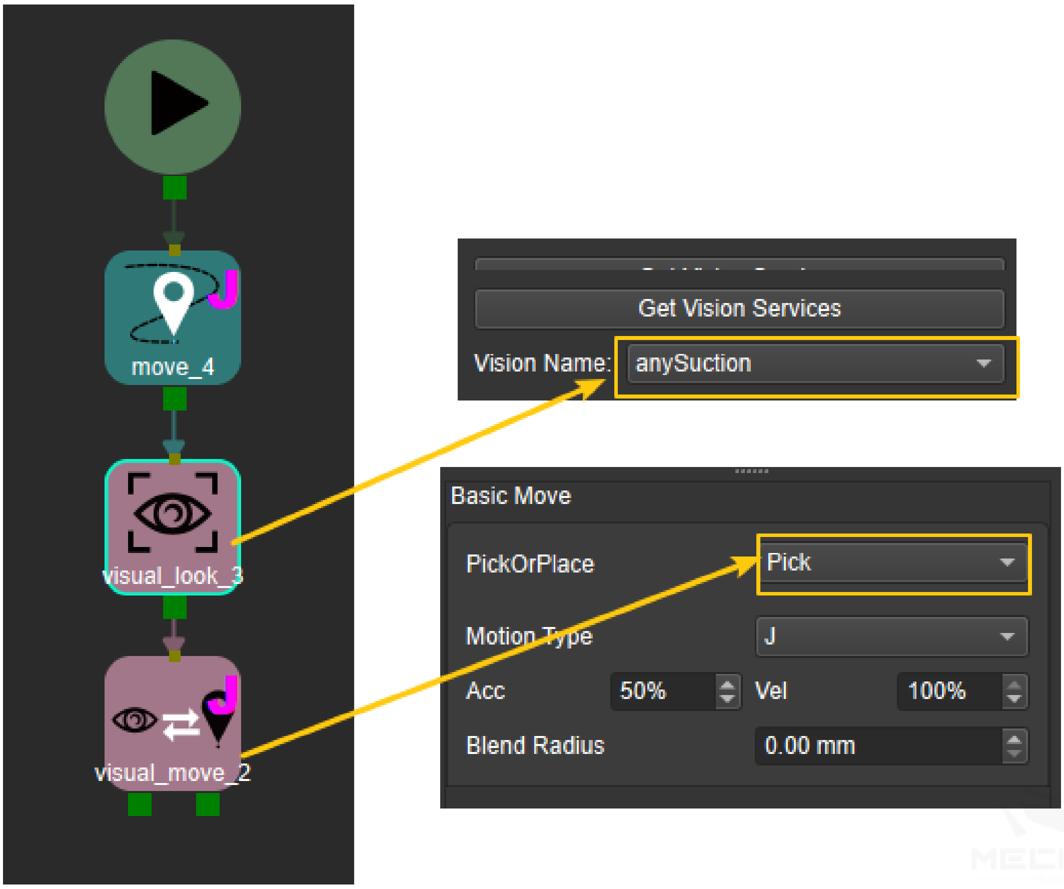
in **Motion Type**) is selected. In case of too many obstacles in the scene, to prevent collisions of the robot with the surrounding objects, which may result in project failure, more move tasks can be added to add intermediate points, thus exerting more control of the robot' s movement.

Introduce Vision Services

After setting the initial pose, vision services can be introduced.

1. As shown in the screenshot, drag a "visual_look" module and a "visual_move" module into the programming interface and connect;
2. For the "visual_look" module' s Name under Property on the right, select the name of a vision service registered in Mech-Vision. For the "visual_move" module, select Pick for PickOrPlace under Basic Move on the right.

When the program runs to the "visual_look" task, it will call the corresponding Mech-Vision project, which takes a picture and return the point cloud and vision results; when the program runs to the "visual_move" task, it takes the vision service results from the most recent "visual_look" task to control the robot to move along the planned trajectory.



Vision Name: anySuction

Basic Move

PickOrPlace: Pick

Motion Type: J

Acc: 50%

Vel: 100%

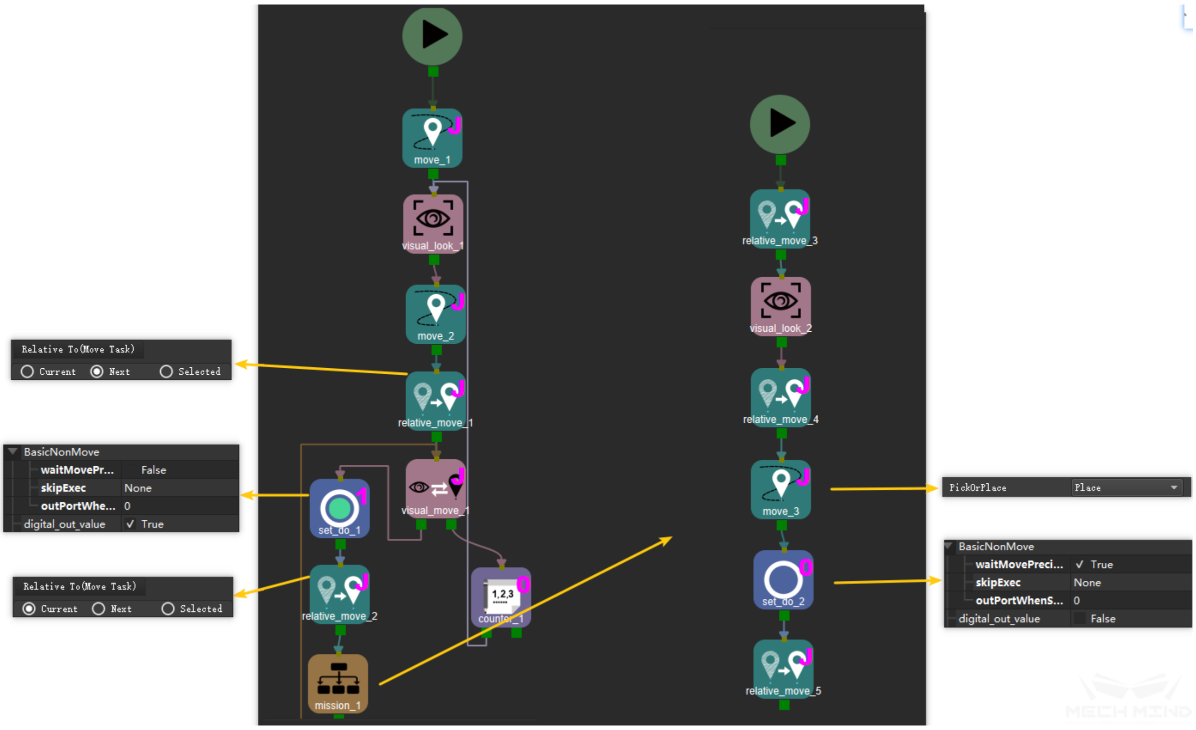
Blend Radius: 0.00 mm

A "visual_move" module has two output ports. When there are vision service results and there is at least one appropriate pick point, the output is made through port 1, and when there is no appropriate pick point, the output is made through port 2.

Build a Complete Project

1. With "visual_look" and "visual_move" tasks as the core of a project, some "move" and "relative_move" tasks can be applied to add intermediate points as appropriate based on the vision service results and the surrounding objects in the scene.
2. Of a "visual_move" module, at port 1, modules for pick-and-place tasks such as palletizing can be added, and at port 2, a "visual_look" module can be added by need to capture images again or modules for other processings can be added.
3. For the mission (the group of modules that collectively serve one particular task, added via "mission" under Topology) that complete picking or placing, the end effector is controlled by adding a "set_do" module that sends digital output.
4. Applications including depalletizing, palletizing, feeding, sorting, etc. are implemented through the cycling of the pick-and-place process described above.

The following figure shows a simple and complete example project. Different modules can be selected to complete the required functions by actual needs.



Relative To(Move Task)
○ Current ● Next ○ Selected

BasicNonMove	
waitMovePr...	False
skipExec	None
outPortWhe...	0
digital_out_value	✓ True

Relative To(Move Task)
● Current ○ Next ○ Selected

PickOrPlace: Place

BasicNonMove	
waitMovePreci...	✓ True
skipExec	None
outPortWhenS...	0
digital_out_value	False

Parameter adjustment and optimization:

- To ensure the readability of the graphical program, please put modules serving one task such as picking into a group, contained by a "mission" module found under Topology in the mission set interface.
- Please add intermediate points as appropriate to prevent collisions or drastic overturning of the robot;
- For a "set_do" module, the entire movement can be smoother by setting Wait Move Precisely to `False` to ready the end effector before the robot reaches the picking pose.

- Before and after picking, by adding a “relative_move” task and setting the Motion Type after picking to L (straight-line move), collisions with surrounding objects, such as the boxes around the target object, can be better avoided.
- If no vision service result is returned or no pick point is found, the visual recognition task should be re-run. If the results are still invalid, the project should be terminated.
- In the pick-and-place mission set, add vision service modules again to take vision results when the robot is placing the object so that the robot can start the next round of pick-and-place missions to shorten the cycle time.
- Taking into consideration the symmetry of the target object, please set PickOrPlace of the “move” task to Place and set the Move Target Type to Obj Pose to remove the relative position binding of the object and the robot TCP。
- Generally, during the move to a fixed point, the speed can be appropriately increased to optimize the cycle time, while the relative movement before and after the picking should be relatively slow to ensure the stability of the picking.

Run the Project

Attention:

- Mech-Viz 1.4.0 will perform a version compatibility check when running the project. It is recommended to use Mech-Vision and Mech-Center 1.4.0 with Mech-Viz 1.4.0.
- If the version of Mech-Vision or Mech-Center used is lower than v1.4.0, a risk warning prompt window will pop up when running the project.

3.3 Open a Project

Open an existing project:

- Click *File* → *Open Project* (shortcut: Ctrl + O). Just select the corresponding folder. There is no need to locate the specific .viz file.
- The Mech-Viz software can be started and the project can be loaded automatically by double-clicking the .viz file.

Open a recently opened project:

Find the project in *File* → *Recent Projects* and open it directly.

Tip:

- If using Mech-Viz v1.4.0 or above to open a project saved with Mech-Viz of versions below v1.4.0, the original .json main file will be automatically saved as and replaced by a .viz file after saving, and will be backed up as a .bak file.
- Click *File* → *Save Project To JSON* to save the project in .json if necessary.

CHAPTER

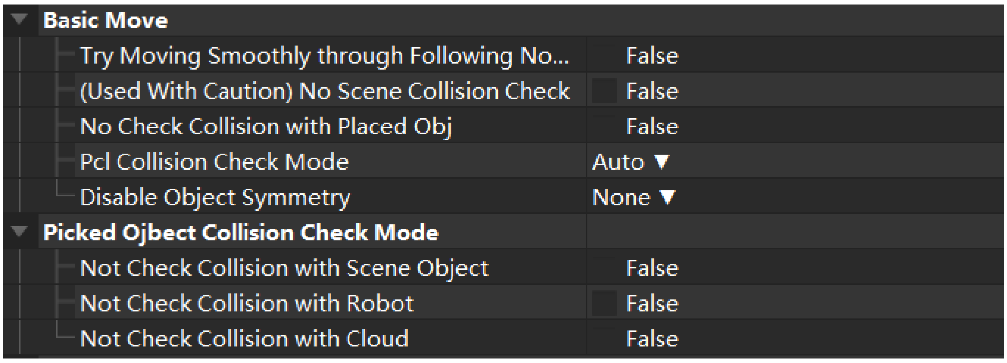
FOUR

SKILLS

4.1 General Parameters of Skills

4.1.1 General Parameters of Move Skills

Hint: Please click on the parameter names below for details.



Basic Move	
Try Moving Smoothly through Following No...	False
(Used With Caution) No Scene Collision Check	False
No Check Collision with Placed Obj	False
Pcl Collision Check Mode	Auto ▼
Disable Object Symmetry	None ▼
Picked Ojbect Collision Check Mode	
Not Check Collision with Scene Object	False
Not Check Collision with Robot	False
Not Check Collision with Cloud	False

Try Moving Smoothly through Following Non-Moves

`False` by default. If set to `True`, when executing the task, the program continues without waiting for the current move task to finish, thus avoiding robot halting and ensuring smoothness of operation.

(Use with Caution) No Scene Collision Check

`False` by default. If set to `True`, when executing the task, the program will not check whether the robot will collide with other objects in the scene and potential damages or injuries may occur. It is recommended to keep the parameter `False`. If it is determined that the robot may have collisions with other objects in the scene, it is forbidden to set the parameter to `True`.

No Collision Check with Placed Obj

`False` by default. If set to `True`, when executing the task, the program will not check whether there will be collisions between the models of the placed object in the scene and the models of the robot and the end effector. The parameter is usually set to `True` for *Movement Tasks* after *Placing* to avoid false collision error alarms due to normal object model overlaps.

PCL Collision Check Mode

Auto by default. For move tasks before the robot picks an object, please set it to NotCheck; for move tasks after the picking, please set to Check. In other cases, normally, this parameter can be kept as default.

Disable Object Symmetry

Please set based on the project. This parameter is only effective for move tasks with object pose as the input parameter, such as the move tasks and pallet tasks with **Move Target Types** set to Obj Pose. This parameter is not effective for move tasks with **Move Target Types** set to TCP Pose or Jps.

If set to True, the robot will strictly place the picked objects based on object poses.

Not Check Collision with Scene Object & Not Check Collision with Robot

If set to True, when executing the task, the program will not check whether there will be collisions between the model of the picked object and the models of the scene objects and the robot to reduce the computation workload of the software and speed up the running, thus shortening the cycle time. This parameter is usually set to True for the one or two move tasks after picking.

Not Check Collision with Cloud

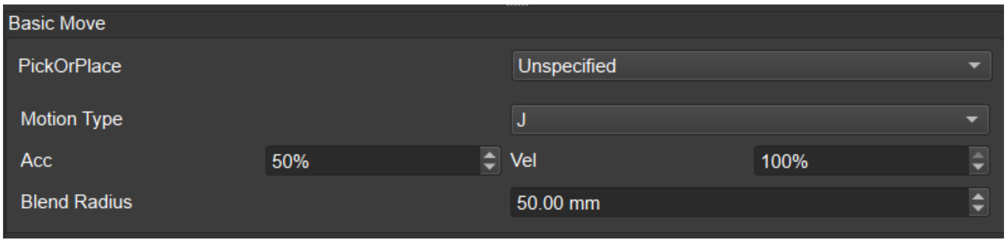
If set to True, when executing the task, the program will not check whether the model of the picked object will collide with the scene point cloud.

1. Setting the parameter to True can help avoid false collision detection alarms.

2. Setting the parameter to True can help reduce the computation workload of Mech-Viz and speed up the running, thus shortening the cycle time.

Basic Move General Parameters

Basic Move general parameters are for adjusting the **speed** and **motion type** of the robot’ s move to a target point.


Basic Move
PickOrPlace: Unspecified
Motion Type: J
Acc: 50%
Vel: 100%
Blend Radius: 50.00 mm

Pick or Place

The pick-or-place setting can be used for Mech-Viz project operation logic check. Please set the corresponding move task to Pick or Place based on the execution logic.

Motion Type

J, i.e., joint motion type is for scenarios where the robot can move within a relatively free spacce and the accuracy requirement for the robot’ s move trajectory is not high.

`L`, i.e., straight line motion type is for scenarios where the accuracy requirement for the robot’ s move trajectory is high, such as welding, gluing, and some types of picking, etc.

Acc & Vel

Please set the **Acc** (acceleration) and **Vel** (velocity) based on the requirements on cycle time, complexities of picking, etc.

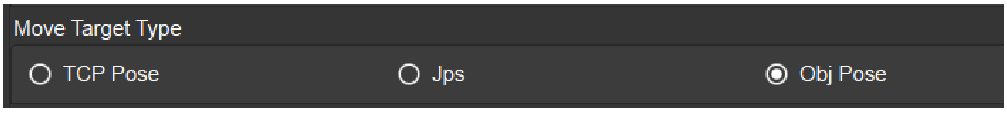
Blend Radius

Blend radius is the radius of turning curves in the robot’ s move trajectory. A higher blend radius means the robot runs more smoothly when making turns.

If the robot moves in a relatively small space, please tune the **Blend Radius** to a smaller value. If the robot moves in a relatively large space without obstacles and two coonsequetive trajectory segments are far away, please tune the **Blend Radius** to a larger value.

Move Target Type

Move target usually should be set to `TCP Pose` or `Jps`. The option `Obj Pose` should be considered only when placing an object.

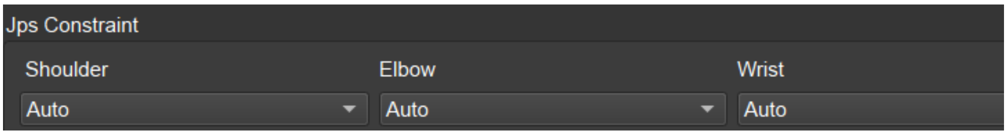


Move Target Type

○ TCP Pose ○ Jps ● Obj Pose

JPs Constraints

The Jps constrains are not for controlling the robot to make shoulder/elbow/wrist flips, but for obtaining the optimal trajectory planning solutions that avoid shoulder/elbow/wrist flips as much as possible.



Jps Constraint

Shoulder	Elbow	Wrist
Auto	Auto	Auto

Other General Parameters

Wait Move Precisely

This parameter is only effective for *visual_look* (`True` by default) and *set_do* (`False` by default).

If set to `True`, the set DO singnal will only be sent to the robot after the robot moves to a specified pose, and in practice the robot will halt for a little while between different move tasks.

If set to `False`, when the robot is able to return a Move ID, or when the robot is not able to return a Move ID but *Use Rough Move ID* in *Others* is selected, the moves before and after this task will be executed without halt and the robot will move smoothly.

Note: During simulation, if some non-move tasks, such as *branch_by_service_message*, *wait_di*, etc., which need to receive external signals, can not be executed onward, please adjust the following two parameters when trasitioning from real execution to simulation for debugging.

Skip Execution

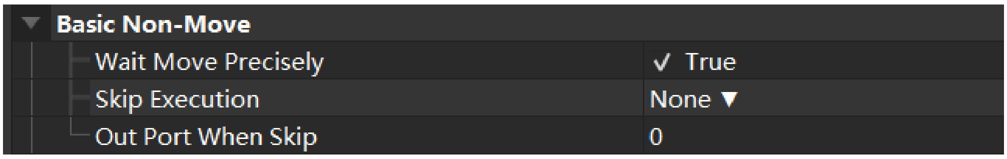
None: default, never skip the task.

WhenSimu: skip the task only during simulation; the out port is specified by **Out Port When Skip**.

Always: always skip the task, whether it is during simulation or actual execution; the out port is specified by **Out Port When Skip**.

Out Port When Skip

This parameter is effective when **Skip Execution** is set to `WhenSimu` or `Alyways`. It sets the out port when skipping a task.



▼ Basic Non-Move	
Wait Move Precisely	✓ True
Skip Execution	None ▼
Out Port When Skip	0

Try Moving Smoothly through Following Non-Moves

Default: `False`

List of values: `True`, `False`

Instruction:

When move tasks are connected through non-move tasks, such as *visual_look*, *set_do*, *check_di*, etc., the robot’ s trajectory planning will be interrupted, and the actual robot will take a short pause, reducing the smoothness of running.

When this parameter is set to `True`, the mission may end prematurely when the move points of a move task are the same as those of the last move task.

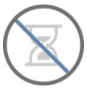
Cause:

When this parameter is set to `True`, multiple move points are sent to the robot at the same time. Mech-Viz considers the robot has finished all the move tasks in sequence if the current JPs sent back is the same as the JPs of the last move point sent to the robot.

Example:

For a trajectory consists of 10 move tasks of which the fifth and the last have the same move point, if the robot’ s speed is low, Mech-Viz may mistakenly determine that the robot has finished the move tasks and prematurely ends the tasks when the robot moves to the move point of the fifth move task, which is the same as that of the last one, and sends the JPs to Mech-Viz.

Display:

Symbol displayed when the parameter is set to `True`: 

(Use with Caution) No Scene Collision Check

Default: `False`
List of values: `False`, `True`

Instructions:
By default, during the running of a Mech-Viz project, when the simulated robot moves along the planned trajectory, Mech-Viz checks whether the robot collides with other models in the scene to determine whether the real robot will collide with surrounding objects.
If the parameter is set to `True`, no such check will be performed and hazards may occur. Please be cautious about setting the parameter to `True`.

No Collision Check with Placed Obj

Default: `False`
List of values: `False`, `True`
Instructions:

When using the function **Detect Collision between Picked Object and Others** in *Picked Object Configuration*, the followings are the two possible cases of errors:

1. In palletizing scenarios, when the robot is placing a carton, the carton to place may come into light contact with the placed cartons (no box deformation will happen). After Mech-Viz detects this collision in simulation, it will plan other places for placing the box, and therefore a full stack can not be formed.

2. Usually, the TCP of a suction cup is not on the surface of it but inside the suction cup model. Under this circumstance, in the simulation of picking, the suction cup may be inside the model of the carton to pick (Mech-Viz does not check the collision between the end effector and the object to pick). After the robot places the object and the carton model turns into an object model in the scene, Mech-Viz will start to check the collision between the suction cup and the carton. Therefore, a warning of detected collision will be issued and the palletizing mission cannot be completed.

When the parameter is set the `True`, no collision check between the robot, end effector, and the placed object will be performed, and the above two cases of errors can be avoided.

Note: Move tasks have the **Pick or Place** setting. Usually, move tasks implemented before *visual_move* should be set to `Pick`, and those after *visual_move* should be set to `Place`. In this way, move tasks after the robot picks the object are placing tasks.

PCL Collision Check Mode

Default: `Auto`
List of values:

`Auto`： default; Mech-Viz only checks point cloud collisions for tasks before and after *visual_move* or *visual_move*.

`NotCheck`： Mech-Viz does not check point cloud collisions for all move tasks.

`Check`： Mech-Viz checks point cloud collisions for all move tasks.

Instructions:

When *Collision Detection Configuration→Detect collision between point cloud and others* is enabled, Mech-Viz will check collisions between the robot, the end effector, and the point cloud when planning the trajectory. Normally, Mech-Viz checks whether the robot collides with other objects during picking and placing. When there are point cloud outliers, non-exiting collisions will be detected, which leads to mistakes in trajectory planning.

Display:
Symbol displayed for `NotCheck`: ◉
Symbol displayed for `Check`: ◉

Disable Object Symmetry

Default: `None`
List of values:
`None`: default; do not disable symmetry on any axis.
`AxisZ`: only disable Z-axis symmetry.
`AxisXY`: only disable X-axis and Y-axis symmetry.
`All`: disable symmetry on all axes.

Instructions:

In some special cases, objects are not pickable due to their peculiar poses. Setting Rotation Symmetry under *End Effector & Objects → Configuration of Objects to Pick (for Move Planning)* may solve this problem. Based on the settings of Rotation Symmetry, when the default object pose is not pickable, Mech-Viz will calculate candidate poses and determine whether the candidate poses are pickable. As the candidate poses calculated based on the settings of Rotation Symmetry are different from the default one, the consistency of the objects’ place poses can not be guaranteed.

Display:
Symbol displayed when set to `AxisZ`: ◔

Symbol displayed when set to `AxisXY`:
Symbol displayed when set to `AxisAll`:

Please see *Symmetry* for more information.

Not Check Collision with Scene Object & Not Check Collision with Robot

Default: `False`
List of values: `False`, `True`
Instructions:
When **Detect collision between picked object and others** under *Collision Detection Configuration → Picked Object Configuration* is enabled, Mech-Viz will check whether the model of the picked object collides with the models of the scene objects and the robot if the two parameters are set to `True`.

Note: In palletizing projects, the calculated carton dimensions have millimeter-level errors, and frictions between cartons may occur during picking but no collisions will happen. For some move tasks that will obviously not cause collisions, checking such collisions only adds to the calculation workload and planning time, and consequently extending the cycle time.

In palletizing projects, setting **Not Check Collision with Scene Object** to `True` does not affect the collision check between the picked carton and the placed cartons. The parameter can be set to `True` when there are scene objects under the stack to avoid failure of finding the palletizing solution.

Not Check Collision with Cloud

Default: `False`
List of values: `False`, `True`
Instructions:

When both **Detect collision between picked object and others** under *Collision Detection Configuration → Picked Object Configuration* and **Detect collision between point cloud and others** under *Collision Detection Configuration → Point Cloud Configuration* are enabled, Mech-Viz will check whether the model of the picked object will collide with the point cloud of the scene.

When Mech-Vision sends the point cloud and object model to Mech-Viz, the point cloud and the object model are fitted together. After the robot picks the object, the model moves along the planned robot trajectory, and the collision between the model of the picked object and the point cloud will occur.

It is known that the model of the picked object will have false collisions with the point cloud. Checking such collisions unnecessarily adds to the calculation workload and extends the planning time of

Mech-Viz.

Pick or Place

Default: `Unspecified`
List of values: `Unspecified`, `Pick`, `Place`
Instructions:

Mech-Viz has a built-in checking mechanism of Mech-Viz project logics. Please set this parameter to `Pick` or `Place` based on the principle of picking before placing.

Normally, please set move tasks before *visual_move* to `Pick` and those after *visual_move* to `Place`.

Motion Type

Default: `J`
List of values: `J` (joint), `L` (line)

Instructions:

The motion types of the robot include joint motion type and straight-line motion type. Joint motion type means that the moving trajectory of the robot consists of curves and is smoother, and robot singularities do not normally occur; straight-line motion type means the moving trajectory of the robot is a straight line, which leads to higher requirement on trajectory planning.

Acc & Vel

Default: the acceleration is 50% by default, the velocity is 100% by default.
Instructions: acceleration and velocity determine the motion velocity of the robot. Normally, the value of acceleration should be lower than that of velocity. Otherwise, the robot will have unpoised movements.

The velocity before and after *visual_move* tasks should be lower to ensure the stability of picking.

Blend Radius

Default: 50.00mm
Instructions:

Blend radius measures the smoothness of the transition segment between two straight-line segments of a trajectory. The larger the value, the smoother the transition.

Usually the parameter can be kept at the default value.

If the robot moves in a relatively narrow space and the blend radius needs to be small, please change this parameter to a smaller value.

If the robot moves in a relatively large space with no barriers, and the distance between two segments of the trajectory is large, this parameter can be changed to a larger value to make the motion transitions smoother.

Move Target Type

1. TCP Pose: The move target point will be represented by the X, Y, Z values and Euler angles or quaternions of the tool coordinate system.
2. JPS: The move target point will be represented by the joint position values of the robot.
3. Obj Pose: The move target point will be represented by the X, Y, Z values and Euler angles or quaternions of the object coordinate system.

The method to determine the move target type is as shown in *Figure 1*.

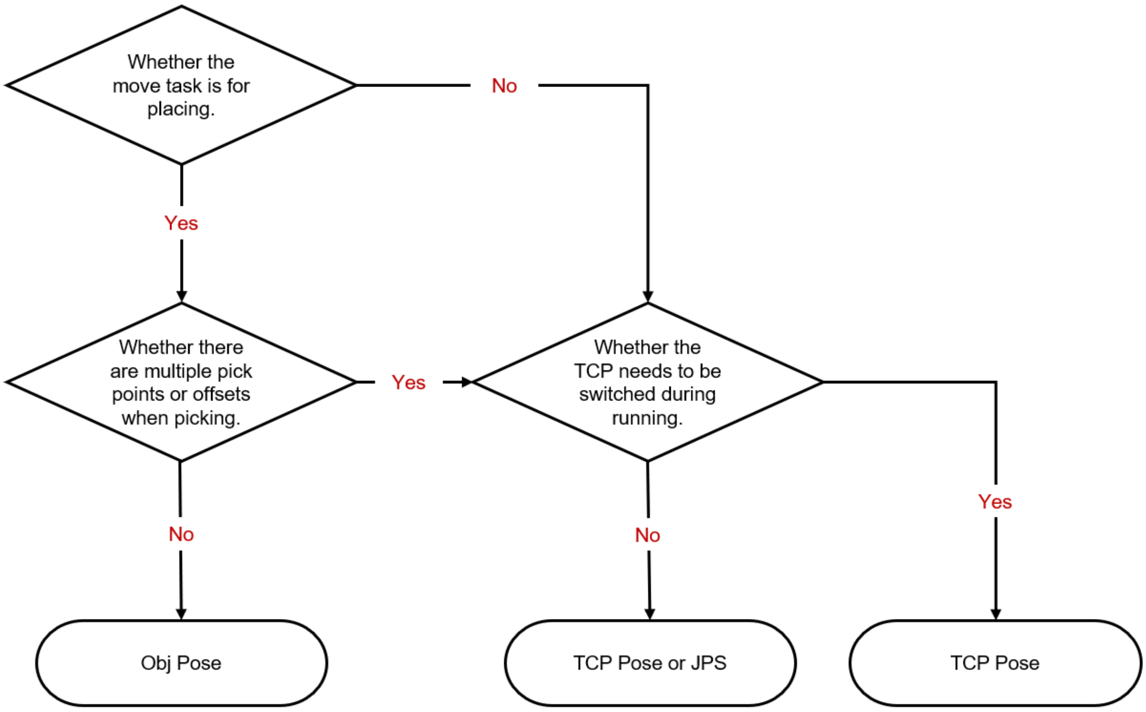
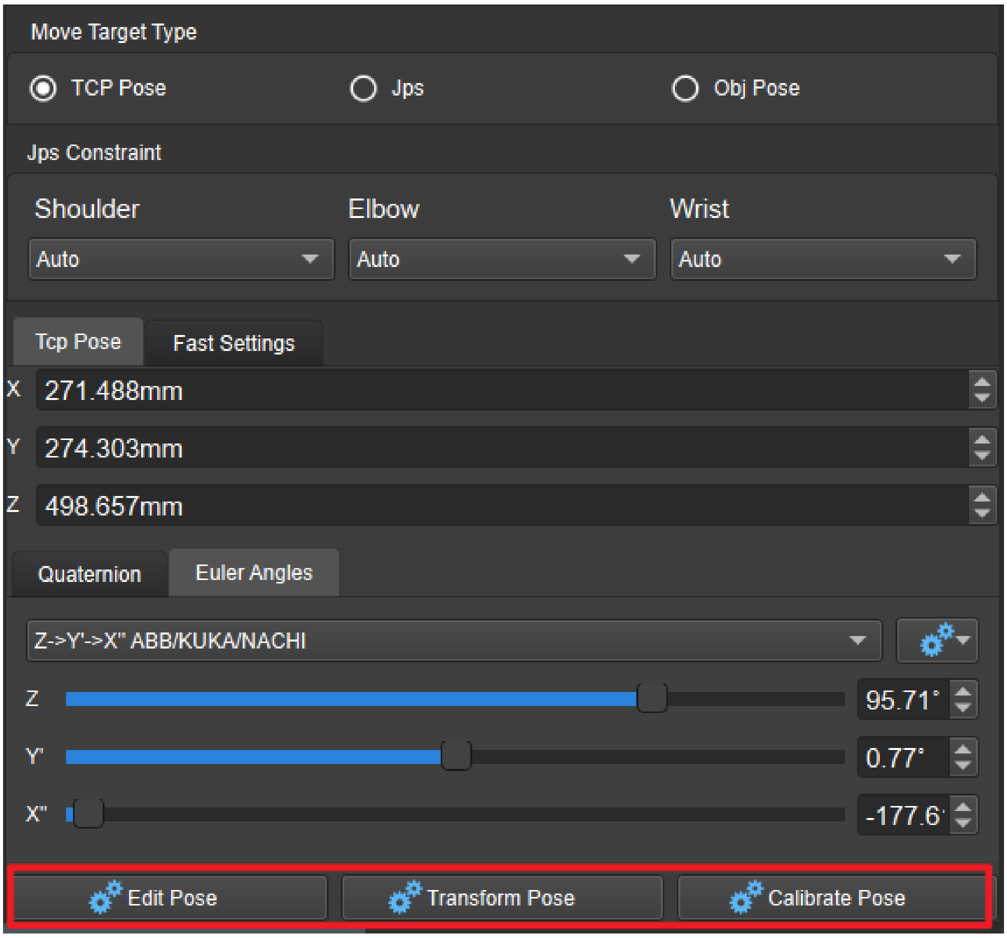

Whether the move task is for placing.
No
Yes
Whether there are multiple pick points or offsets when picking.
Yes
Whether the TCP needs to be switched during running.
No
No
Yes
Obj Pose
TCP Pose or JPS
TCP Pose

Figure 1. Method to determine the move target type

- **Shortcuts for adjusting TCP Pose & Obj Pose**



Move Target Type

● TCP Pose ○ Jps ○ Obj Pose

Jps Constraint

Shoulder: Auto

Elbow: Auto

Wrist: Auto

X 271.488mm

Y 274.303mm

Z 498.657mm

Z->Y'->X" ABB/KUKA/NACHI

Z 95.71°

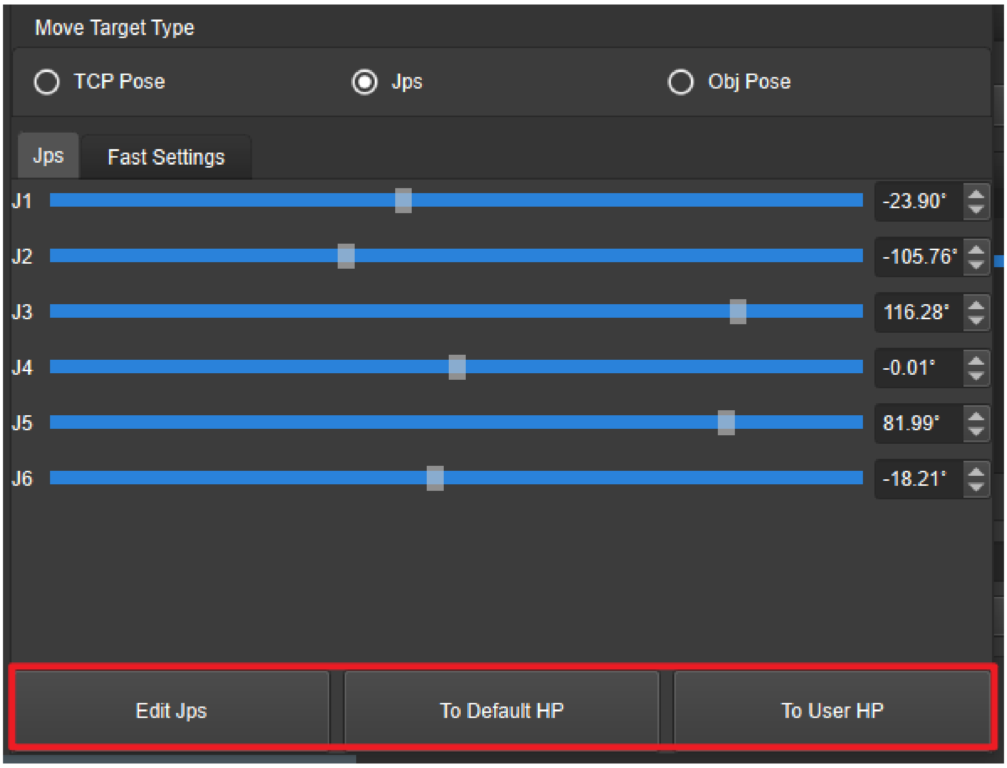
Y' 0.77°

X" -177.6

Edit Pose: Edit the pose directly by copying and pasting or editing the X, Y, Z coordinate values and the quaternion.
Transform Pose: Adjust the robot pose; the reference coordinate system can be either the tool coordinate system of the current robot pose (flange coordinate system) or that of the default robot pose (robot base coordinate system). It is suitable for fine-tuning.
Calibrate Pose: The calibration method is similar to the three-point method of ABB Robotics in calculating the workpiece coordinate system. It is suitable for cases where the objects are prone to rotation and the object poses can not be easily determined. For instance, when a cuboid tilts, its pose is hard to determine. Calibration pose can be used here to calculate the cuboid' s pose and therefore the robot can run based on the calculated pose.

Hint: Click the corresponding button to enter the pop-up window for related settings.

- **Shortcuts for adjusting JPS**



Move Target Type

○ TCP Pose ● Jps ○ Obj Pose

J1 -23.90°

J2 -105.76°

J3 116.28°

J4 -0.01°

J5 81.99°

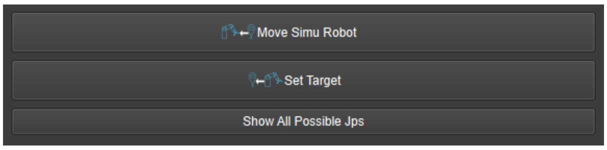
J6 -18.21°

Edit JPS: Similar to **Edit Pose** above; edit the JPs by copying and pasting or editing the JPs in the form of either radians or degrees, which can be switched based on the actual needs.

To Default HP: The default home position is the zero moment point of the robot. This function enables the robot to return to the zero moment point.

To User HP: The user home position is set under *Robot→Joint Positions*. This function enables the robot to return to the home position quickly. If the user home position is not set, it will be the same as the default home position.

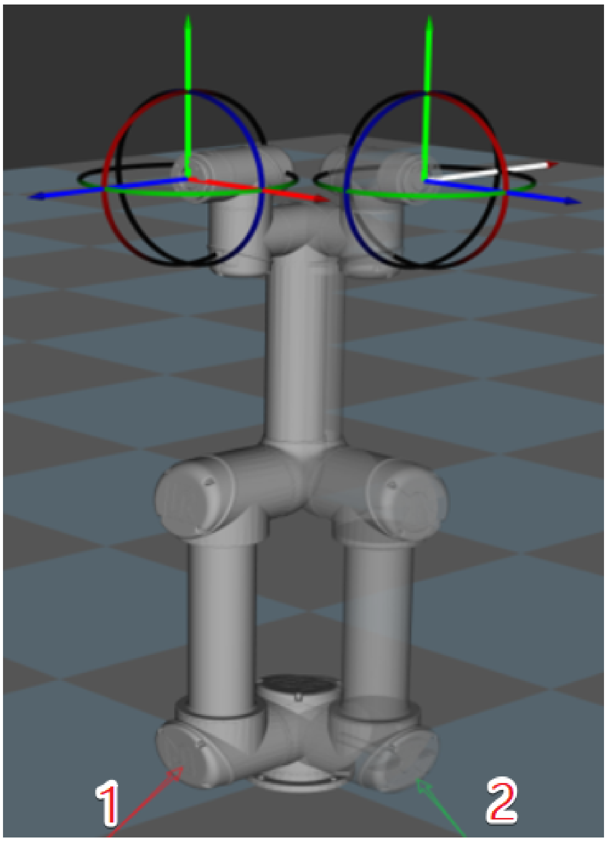
- **Fast Settings**



Move Simulated Robot: Move the simulated robot model to the current position of the real robot, that is, move the simulated robot model from position 1 to position 2. Only the position of the simulated robot is changed and the position of the real robot will not be changed.

Set Target: Set the move target point to the position of the simulated robot model, that is, set the move target point from position 1 to position 2. The position of the simulated robot is not changed, while the move target point is changed.

Show all possible JPS' : Show all the JPs solutions to the current move target point; the maximum number of solutions is eight. Click *Move Simulated Robot* to move the robot to the pose of the optimal solution.



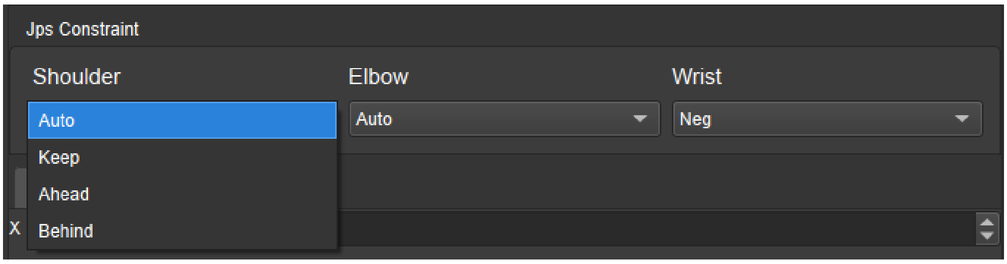
JPs Constraints

Definition of Terms

Shoulder: The relative positional relationship between the center of the wrist and Axis1. Axis1 refers to the rotation center axis of the robot axis 1.

Elbow: Relative positional relationship between wrist and LowerArm. LowerArm refers to the line connecting the rotation centers of the robot axis 2 and axis 3.

Wrist: Robot axis 5 is the wrist. Whether the wrist JPs can be negative means whether the robot's wrist can rotate in the negative direction.



Jps Constraint

Shoulder: Auto (Auto, Keep, Ahead, Behind)

Elbow: Auto

Wrist: Neg

Figure 1. JPs constraint settings

List of Values

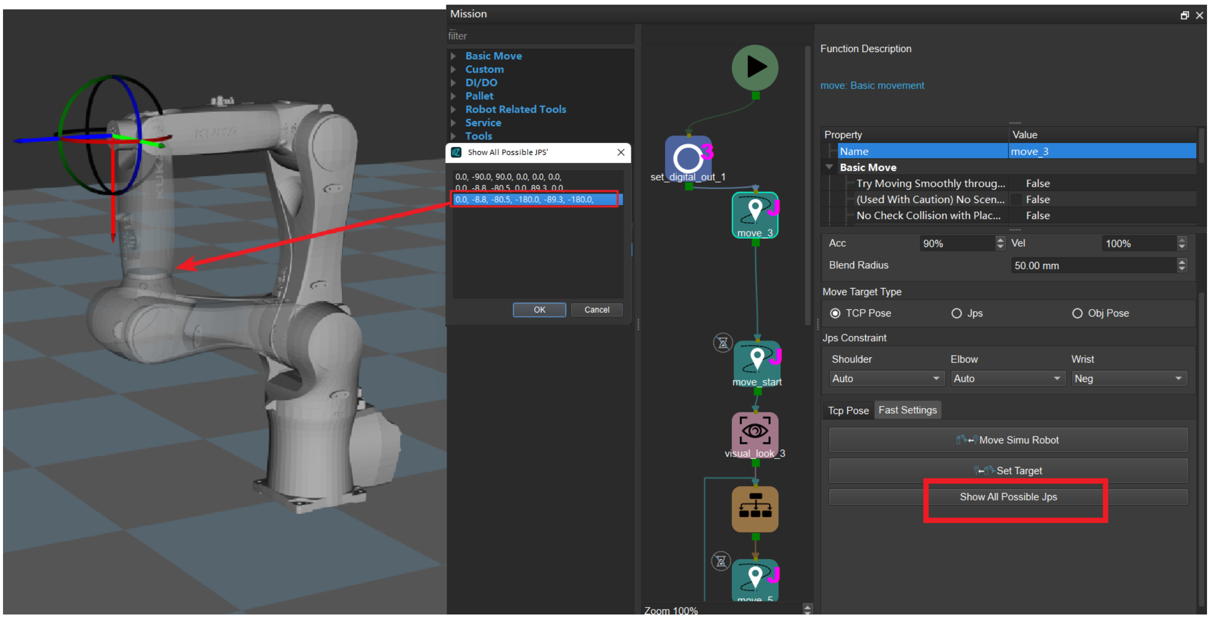
Auto: This joint’s motion is under completely zero constraints. The optimal solution is the one in which each axis undergoes the minimum amount of rotation.

Keep: Record the current JPs status of the robot as the constraints for getting the next JPs solution. Taking axis 3 as an example, if axis 3 is positive when getting the next JPs solution, only solutions in which axis 3 is positive will be considered.

Ahead: The wrist center is ahead of Axis1.

Behind: The wrist center is behind Axis1.

After clicking *Fast Settings → Show All Possible JPS’*, a window showing all the JPs solutions to the current target point will pop up. After clicking one of the solutions, the corresponding pose of the solution will be displayed in the simulation, thus showing the possible solutions under different constraints, as shown in *Figure 2*



Show All Possible JPS'

```
0.0, -90.0, 90.0, 0.0, 0.0, 0.0,
0.0, -8.8, -80.5, 0.0, 89.3, 0.0,
0.0, -8.8, -80.5, -180.0, -89.3, -180.0,
```

Function Description: move: Basic movement

Property	Value
Name	move_3
Basic Move	
Try Moving Smoothly throug...	False
(Used With Caution) No Scen...	False
No Check Collision with Plac...	False

Acc: 90% Vel: 100%

Blend Radius: 50.00 mm

Move Target Type: ● TCP Pose ○ Jps ○ Obj Pose

Jps Constraint: Shoulder: Auto Elbow: Auto Wrist: Neg

Figure 2. Viewing possible JPs solutions

Attention:

1. JPs constraint settings only apply to 6-axis robots. 4-axis robots are considered as not having abnormal shoulder, elbow, and wrist positions.
2. relative_move, smart_pallet_pattern, and mixed_pallet do not support JPs constraint settings and have default JPs constraints, that is, the statuses of the shoulder, elbow, and wrist do not change, the robot’ s motion does not cross solution systems (does not go through singularities).

4.2 Basic Move

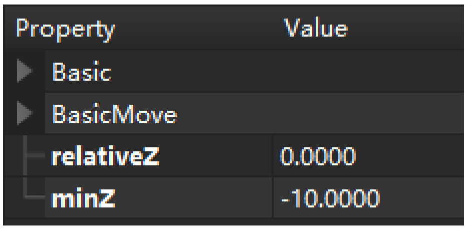
Mech-Viz motion module is a basic module that controls the robot to complete various actions. By setting the parameters, different brands of robots can achieve the desired motion effect.

4.2.1 dynamic_move

Description

Used in eye in hand mode, the initial pose is the highest pose, and the subsequent motion changes dynamically with the height of the object in the field of view.

Parameters



Property	Value
▶ Basic	
▶ BasicMove	
relativeZ	0.0000
minZ	-10.0000

BasicMove See *General Parameters of Move Skills*

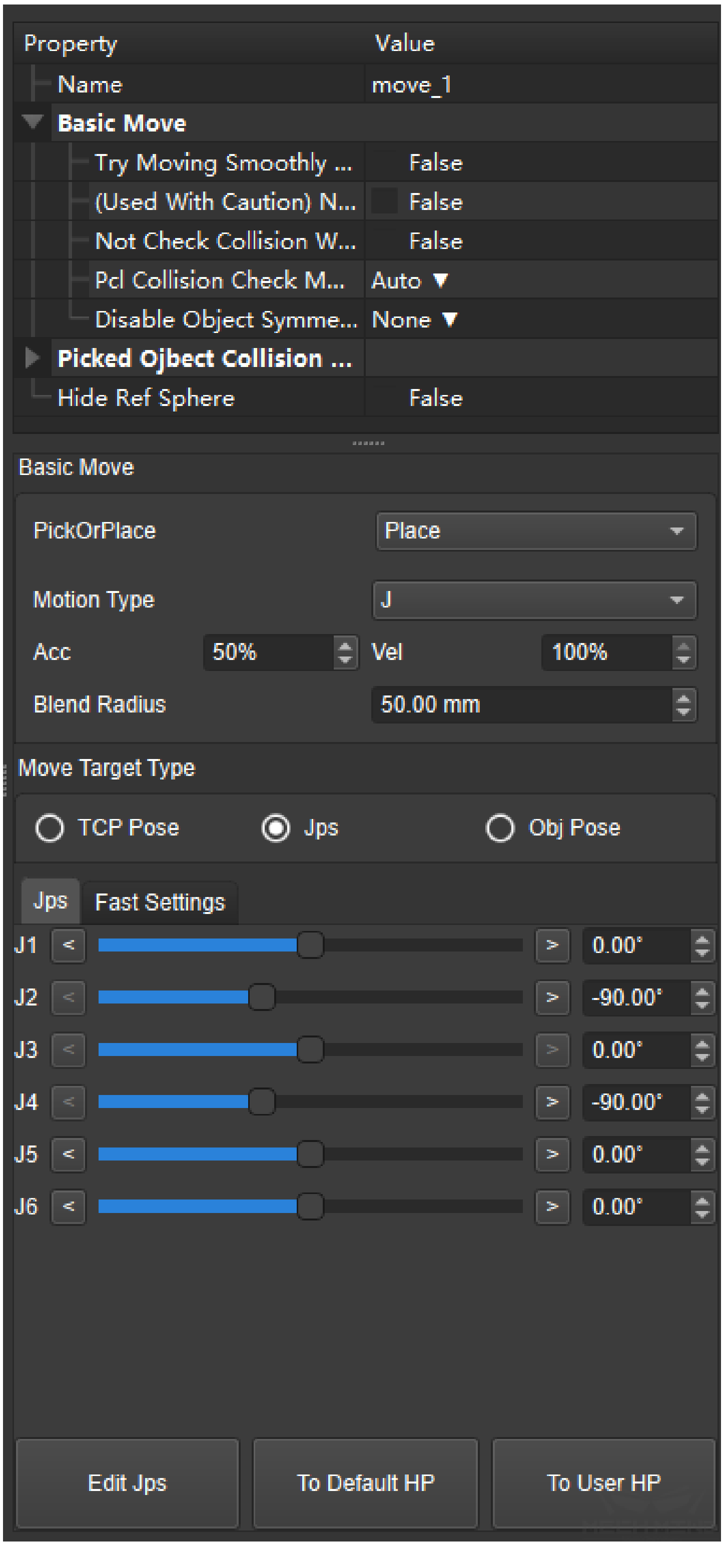
relativeZ Based on the height of the highest object acquired by vision, it will calculate the height required for this movement according to this parameter. This height cannot be greater than the height of the initial pose, and the minimum cannot be less than *minZ*.

minZ The minimum height in Z direction that the robot is actually allowed to reach.

Basic Move See *Basic Move General Parameters*

4.2.2 move

Description



Property	Value
Name	move_1
Basic Move	
Try Moving Smoothly ...	False
(Used With Caution) N...	False
Not Check Collision W...	False
Pcl Collision Check M...	Auto ▼
Disable Object Symme...	None ▼
Picked Ojbect Collision ...	
Hide Ref Sphere	False

Basic Move

PickOrPlace: Place

Motion Type: J

Acc: 50% Vel: 100%

Blend Radius: 50.00 mm

Move Target Type

○ TCP Pose ● Jps ○ Obj Pose

Jps	Fast Settings
J1	0.00°
J2	-90.00°
J3	0.00°
J4	-90.00°
J5	0.00°
J6	0.00°

Parameters of Move

Set a target pose in the robot’ s motion path and the way to move to that pose

Parameters

BasicMove See *General Parameters of Move Skills*

hideRefSphere Hide reference ball model.

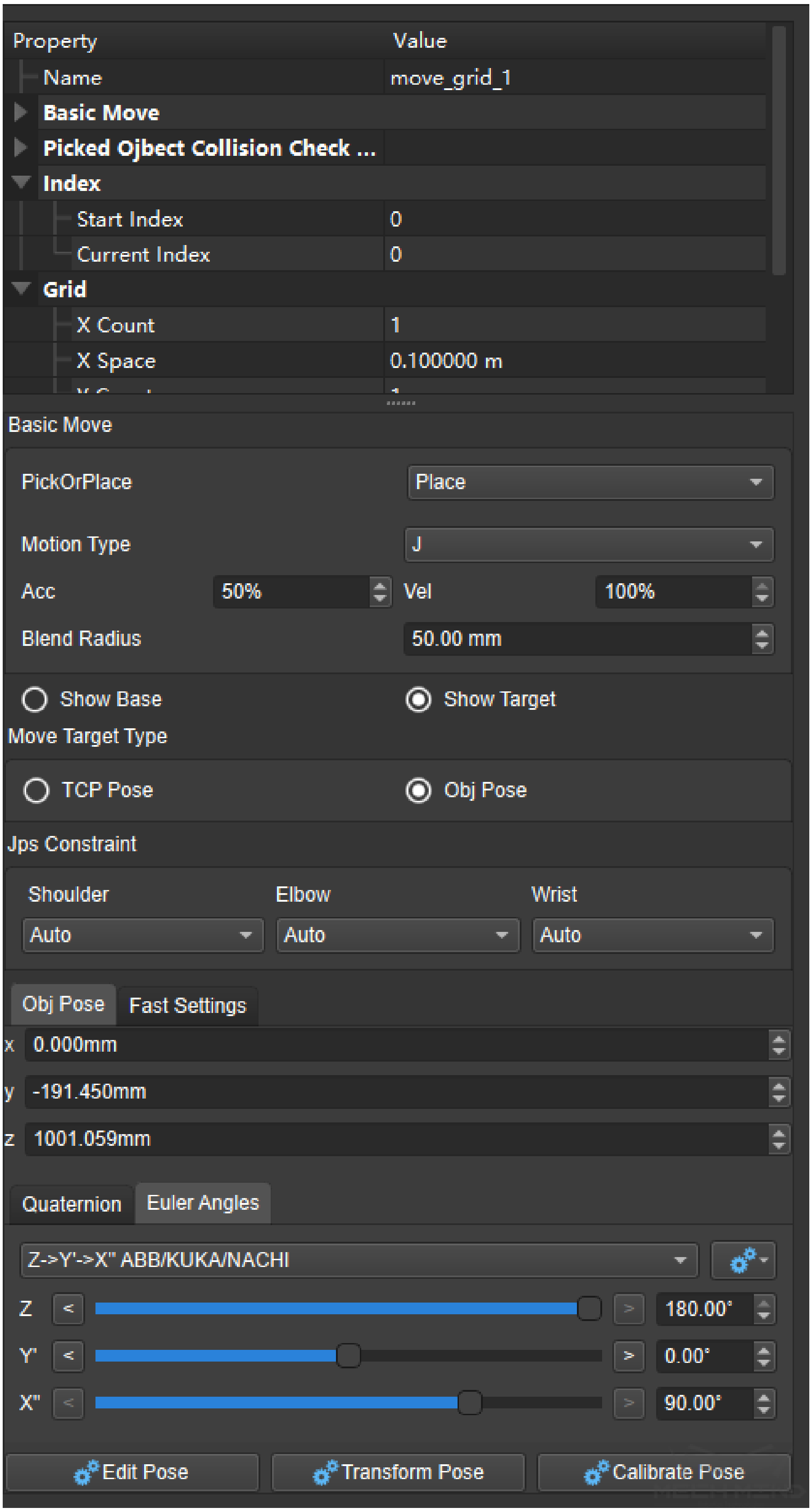
Offset X/Y/Z offset: if the movement type is TCP Pose or Obj Pose, it is the X/Y/Z offset relative to the specified point in the corresponding reference system.

Example: if the Z-direction offset is 0.2 m, when it is TCP Pose, the actual position of the robot’ s movement is the position where the specified point extends 0.2 m in the positive direction along the z-axis of the TCP coordinate system.

Basic Move See *Basic Move General Parameters*

4.2.3 move_grid

Description



Property	Value
Name	move_grid_1
Basic Move	
Picked Ojbect Collision Check ...	
Index	
Start Index	0
Current Index	0
Grid	
X Count	1
X Space	0.100000 m

Basic Move

PickOrPlace: Place

Motion Type: J

Acc: 50% Vel: 100%

Blend Radius: 50.00 mm

○ Show Base ● Show Target

Move Target Type

○ TCP Pose ● Obj Pose

Jps Constraint

Shoulder	Elbow	Wrist
Auto	Auto	Auto

Obj Pose | Fast Settings

x 0.000mm

y -191.450mm

z 1001.059mm

Quaternion | Euler Angles

Z->Y'->X" ABB/KUKA/NACHI

Z 180.00°

Y' 0.00°

X" 90.00°

move_grid

Move the points in a specified array

Parameters

Basic parameters of movement tasks See *General Parameters of Move Skills*

Index

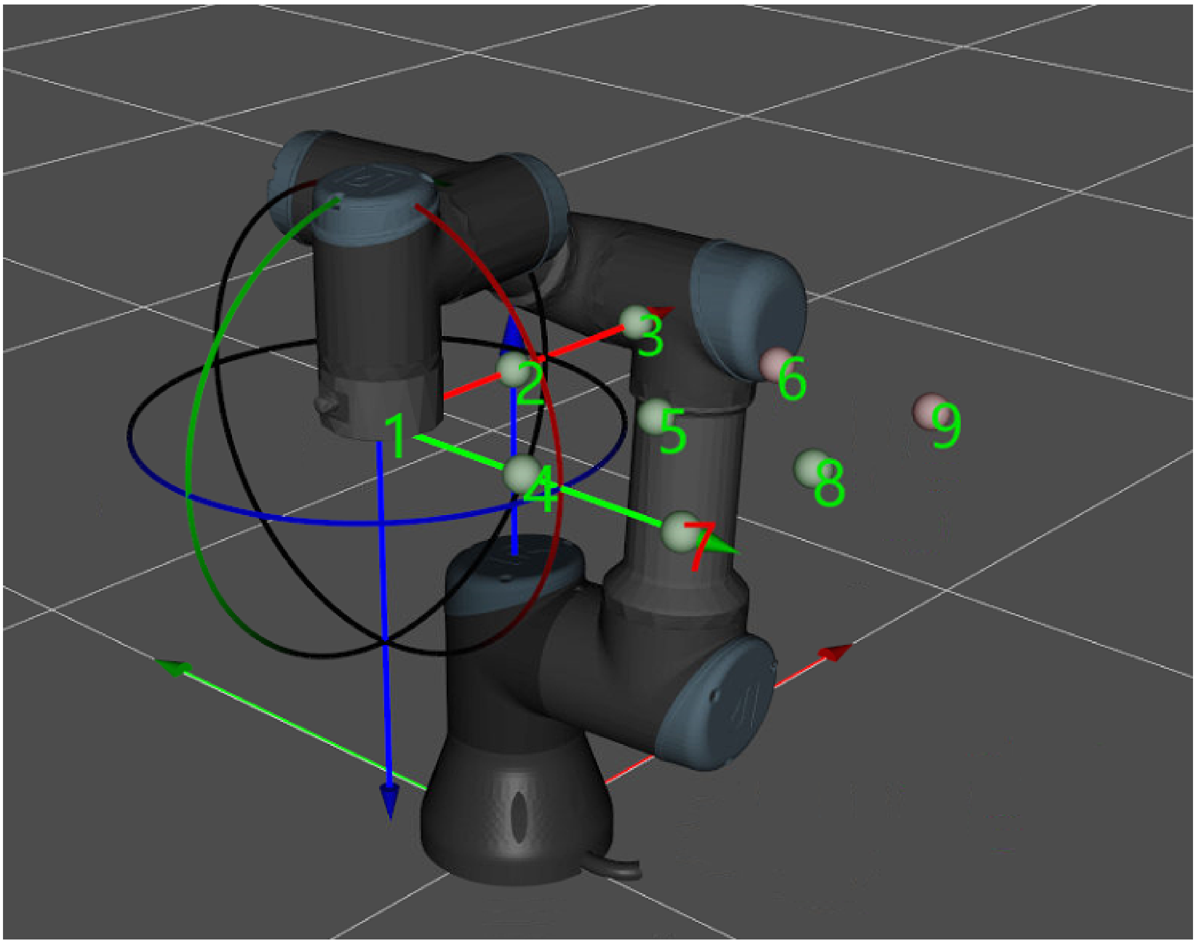
- **startIndex**: array from a specified starting point.
- **curIndex**: the number of points in the current array.

Grid

- **xCount**: the number of points in the X direction of the array.
- **xSpace**: the interval between every two points in the X direction of the array.
- **yCount**: the number of points in the Y direction of the array.
- **ySpace**: the interval between every two points in the Y direction of the array.
- **zCount**: the number of points in the Z direction of the array.
- **zSpace**: the interval between every two points in the Z direction of the array.

Basic Move See *Basic Move General Parameters*

Note: Select Show Base. Array points and the number of each point are shown in Mech-Viz, as shown in the figure:

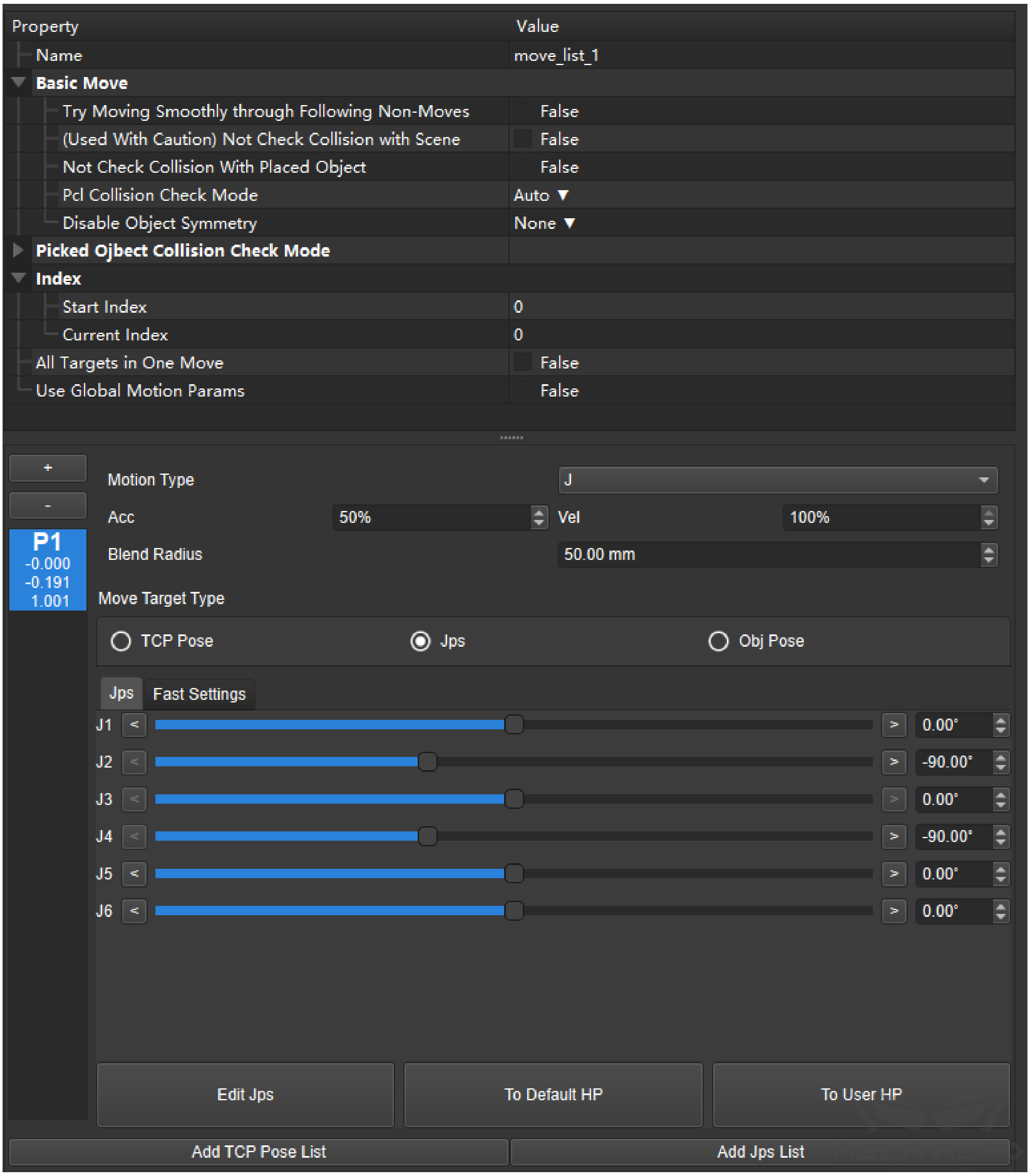


Show target point grid

Base Pose Determine the pose of the array starting point by X, Y, Z coordinates and quaternions or Euler angles (the overall array will rotate with it).

4.2.4 move_list

Description



Property	Value
Name	move_list_1
Basic Move	
Try Moving Smoothly through Following Non-Moves	False
(Used With Caution) Not Check Collision with Scene	False
Not Check Collision With Placed Object	False
Pcl Collision Check Mode	Auto ▼
Disable Object Symmetry	None ▼
Picked Ojbect Collision Check Mode	
Index	
Start Index	0
Current Index	0
All Targets in One Move	False
Use Global Motion Params	False

P1
-0.000
-0.191
1.001

Motion Type: J

Acc: 50%

Vel: 100%

Blend Radius: 50.00 mm

Move Target Type

○ TCP Pose ● Jps ○ Obj Pose

J1	0.00°
J2	-90.00°
J3	0.00°
J4	-90.00°
J5	0.00°
J6	0.00°

move_list

Move by the points in a specified sequence

Parameters

Basic & BasicMove See *General Parameters of Move Skills*

Index

- **startIndex**: array from a specified starting point.
- **curIndex**: the number of points in the current array.

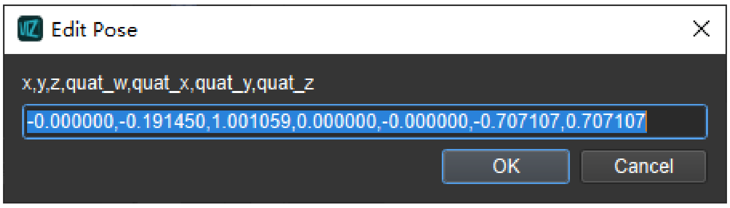
move_list When it is checked as True, all motion points in the sequence will be gone through at one time.

visionTrajPose When it is checked as True, it is considered as a visually planned path sequence. This function is often used for path scenarios such as glue coating.

useGlobalMotionParams Whether all motion points in the path use global motion parameters.

Basic Move See *Basic Move General Parameters*

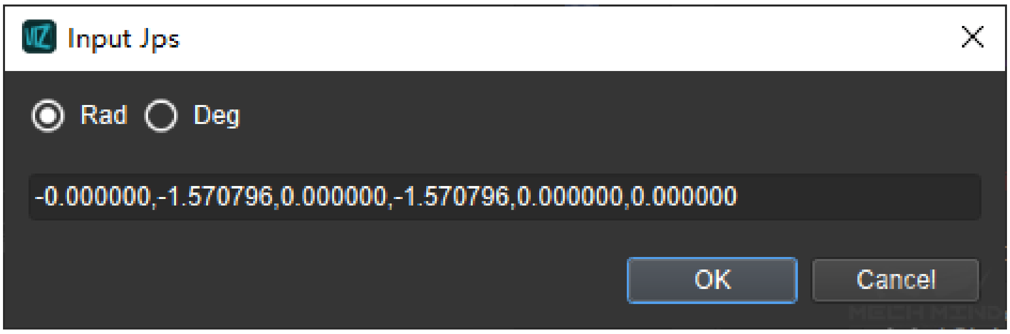
Add TCP Pose As shown in the figure below, inputting the tool pose by rows can add a new movement point.



x,y,z,quat_w,quat_x,quat_y,quat_z

-0.000000,-0.191450,1.001059,0.000000,-0.000000,-0.707107,0.707107

Add Jps As shown in the figure below, inputting the joint angle by rows can add a new movement point.



● Rad ○ Deg

-0.000000,-1.570796,0.000000,-1.570796,0.000000,0.000000

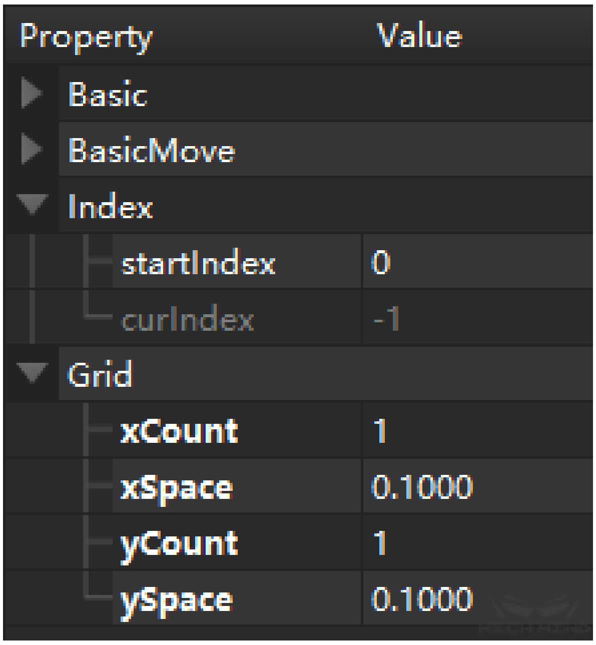
4.2.5 outer_index_move

Description

Move the points in array based on the set grid parameters of X/Y, and combined with the index value obtained from the external service and the position of this point. It must be used with adapter

The array points that robot can move are more flexible than *move_grid*, it is no need to move according to the regular n × n array.

Parameters



Property	Value
Basic	
BasicMove	
Index	
startIndex	0
curIndex	-1
Grid	
xCount	1
xSpace	0.1000
yCount	1
ySpace	0.1000

BasicMove See *General Parameters of Move Skills*

index

- **startIndex**: array from a specified starting point.
- **curIndex**: the number of points in the current array. It can be changed by adapter.

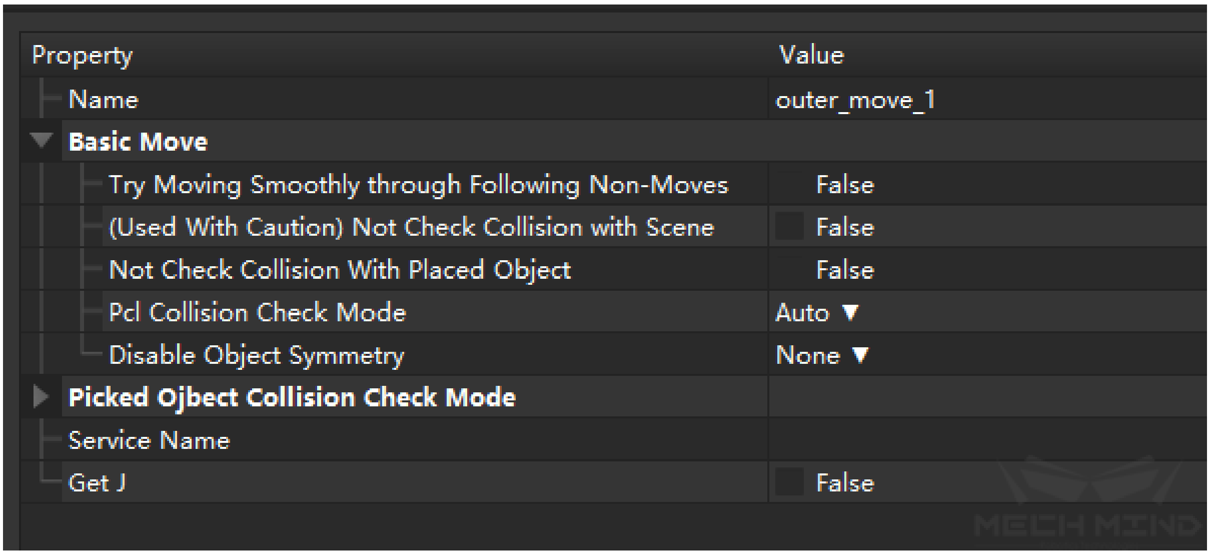
Grid

- **xCount**： the number of points in the X direction of the array.
- **xSpace**： the interval between every two points in the X direction of the array.
- **yCount**： the number of points in the Y direction of the array.
- **xSpace**： the interval between every two points in the Y direction of the array.

Basic Move See *Basic Move General Parameters*

4.2.6 outer_move

Description



Property	Value
Name	outer_move_1
Basic Move	
Try Moving Smoothly through Following Non-Moves	False
(Used With Caution) Not Check Collision with Scene	□ False
Not Check Collision With Placed Object	False
Pcl Collision Check Mode	Auto ▼
Disable Object Symmetry	None ▼
Picked Ojbect Collision Check Mode	
Service Name	
Get J	□ False

Perform target pose to be moved obtained from external services

This module is used by Mech-Viz to obtain the target pose moved from an external service and run the robot to that pose. It must be used in conjunction with the adapter

Parameters

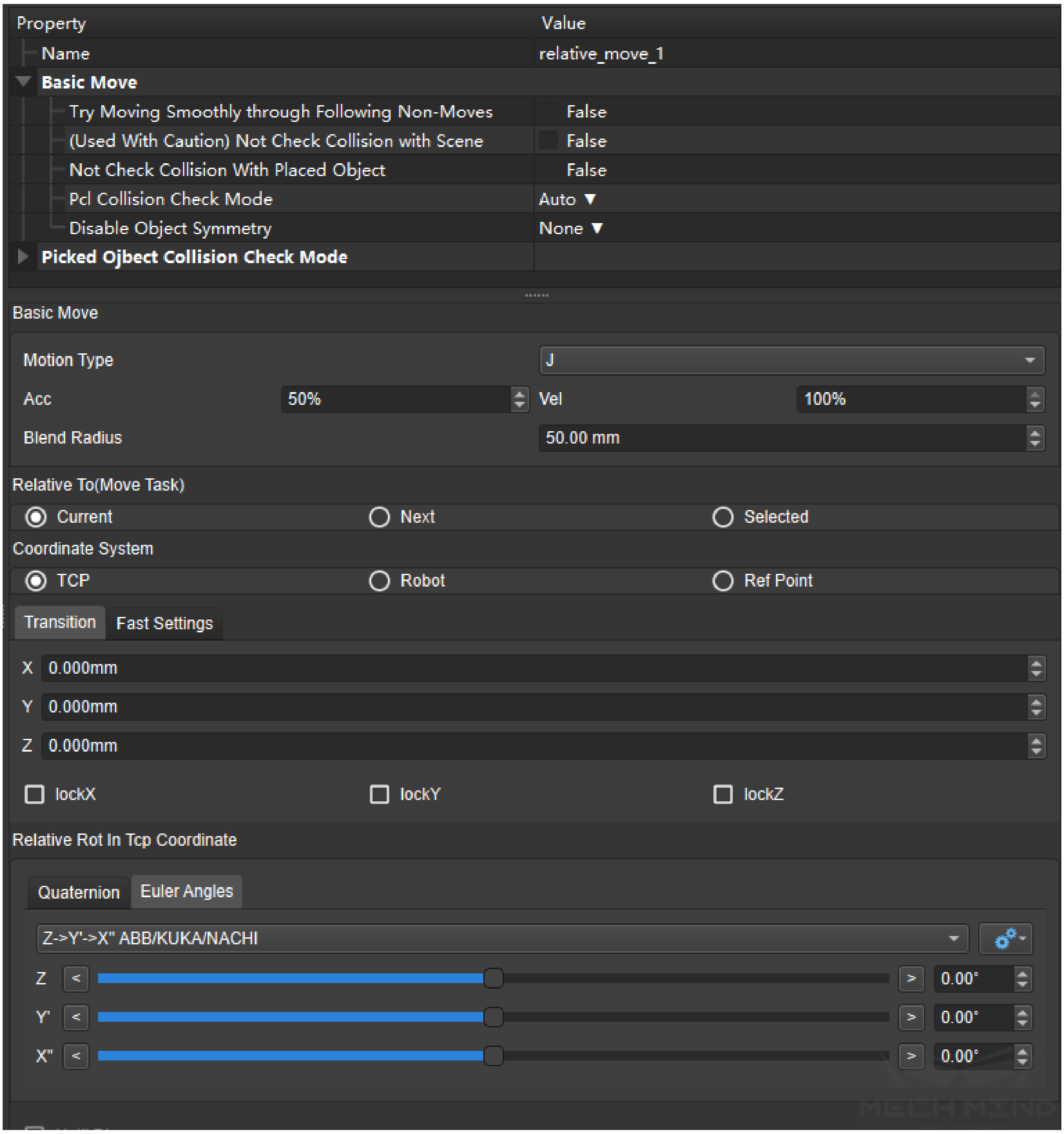
Basic parameters of movement tasks See *General Parameters of Move Skills*

serviceName The external server name registered by the adapter on the Mech-Center. This parameter must be unified with the adapter to obtain the module interface via the server name and send to the destination.

getJ By default it is False, the position where the last planning ended will be used for the initial position of the software planning trajectory; if it is checked as True, the initial position of the software planning trajectory will be updated to the robot joint position obtained from the external. It is usually used when Mech-Viz does not fully control the robot motion

4.2.7 relative_move

Description



Property	Value
Name	relative_move_1
Basic Move	
Try Moving Smoothly through Following Non-Moves	False
(Used With Caution) Not Check Collision with Scene	False
Not Check Collision With Placed Object	False
Pcl Collision Check Mode	Auto ▼
Disable Object Symmetry	None ▼
Picked Ojbect Collision Check Mode	

Basic Move

Motion Type: J

Acc: 50% Vel: 100%

Blend Radius: 50.00 mm

Relative To(Move Task)

● Current ○ Next ○ Selected

Coordinate System

● TCP ○ Robot ○ Ref Point

Transition | Fast Settings

X 0.000mm

Y 0.000mm

Z 0.000mm

□ lockX □ lockY □ lockZ

Relative Rot In Tcp Coordinate

Quaternion | Euler Angles

Z->Y'->X" ABB/KUKA/NACHI

Z 0.00°

Y' 0.00°

X" 0.00°

relative_move

Relative movement based on the known motion points

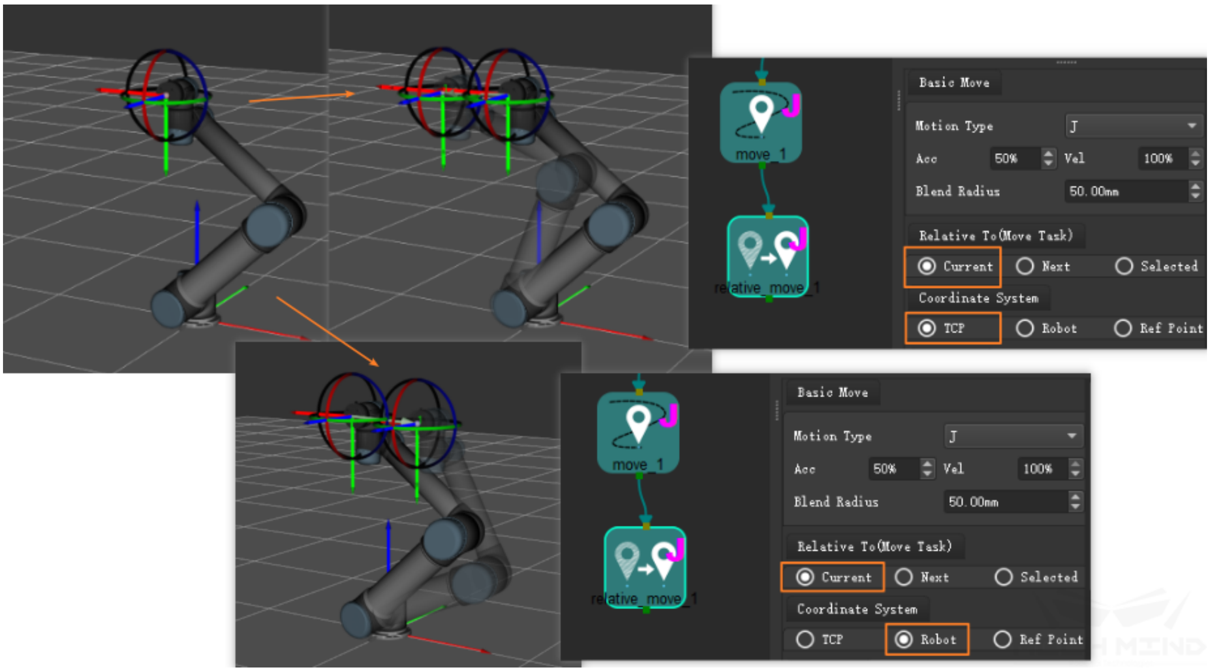
Parameters

Basic & Basic Move See *General Parameters of Move Skills*

Basic Move See *Basic Move General Parameters*

Relative to (movement tasks) The relative position setting includes the setting of the reference point, the relative coordinate system, and the offset vector. Wherein the reference point can be *Current* , or you can choose the *Next point* in the track. The coordinate system where the offset vector is located can be *TCP* , or *Robot* , and *Ref Point* . The offset vector is expressed by x, y, and z.

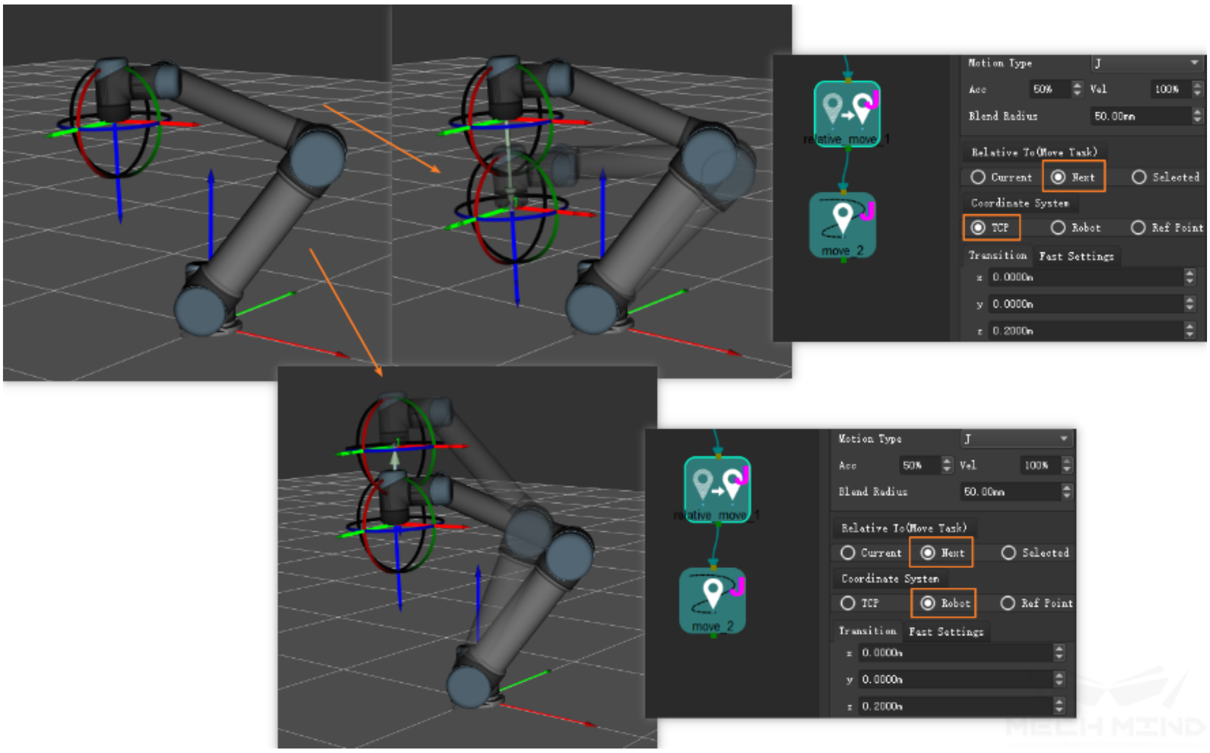
If you choose *TCP* , robot will relative move based on the TCP coordinate system; and if you choose *Robot* , robot will relative move based on the Base coordinate system. The comparison between moving based on two coordinate systems shown in the figure below:



Basic Move
Motion Type: J
Acc: 50% Vel: 100%
Blend Radius: 50.00mm
Relative To(Move Task): ● Current ○ Next ○ Selected
Coordinate System: ● TCP ○ Robot ○ Ref Point

Basic Move
Motion Type: J
Acc: 50% Vel: 100%
Blend Radius: 50.00mm
Relative To(Move Task): ● Current ○ Next ○ Selected
Coordinate System: ○ TCP ● Robot ○ Ref Point

relative to current

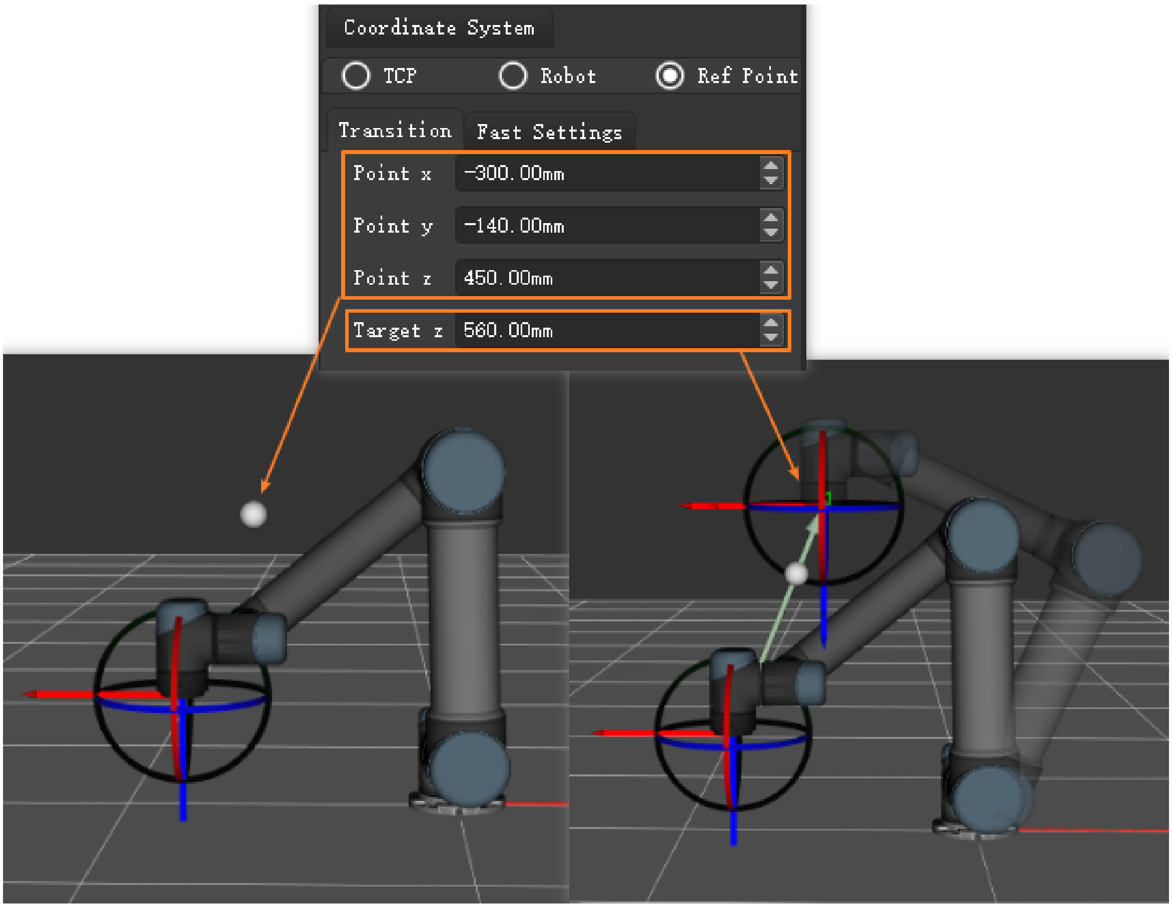


Motion Type: J
Acc: 50% Vel: 100%
Blend Radius: 50.00mm
Relative To(Move Task): ○ Current ● Next ○ Selected
Coordinate System: ● TCP ○ Robot ○ Ref Point
Transition | Fast Settings
x: 0.0000m
y: 0.0000m
z: 0.2000m

Motion Type: J
Acc: 50% Vel: 100%
Blend Radius: 50.00mm
Relative To(Move Task): ○ Current ● Next ○ Selected
Coordinate System: ○ TCP ● Robot ○ Ref Point
Transition | Fast Settings
x: 0.0000m
y: 0.0000m
z: 0.2000m

relative to next (point)

Reference point description When the detected objects are in deep box, you can choose *Ref Point* to avoid collision of end effector and box during vertical picking(default Z direction of relative_move), relative_move will use the *Ref Point* as the target direction, and the movement distance will be the preset value.



Coordinate System

○ TCP ○ Robot ● Ref Point

Transition | Fast Settings

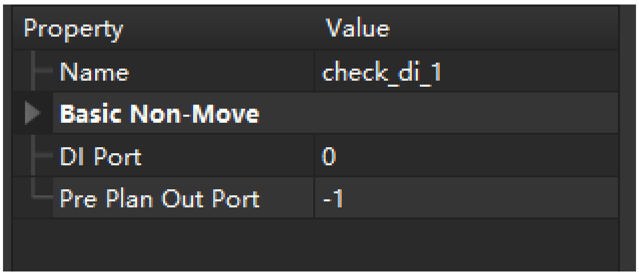
Point x	-300.00mm
Point y	-140.00mm
Point z	450.00mm
Target z	560.00mm

relative_move when Coordinate System is Ref Point

4.3 DI/DO

4.3.1 check_di

Description



Property	Value
Name	check_di_1
Basic Non-Move	
DI Port	0
Pre Plan Out Port	-1

Detect the signal value of the specified DI port of the robot. There are two output ports, if the value is

0, this task will output from left port, otherwise from right port.

Parameters

BasicNonMove See *Other General Parameters*

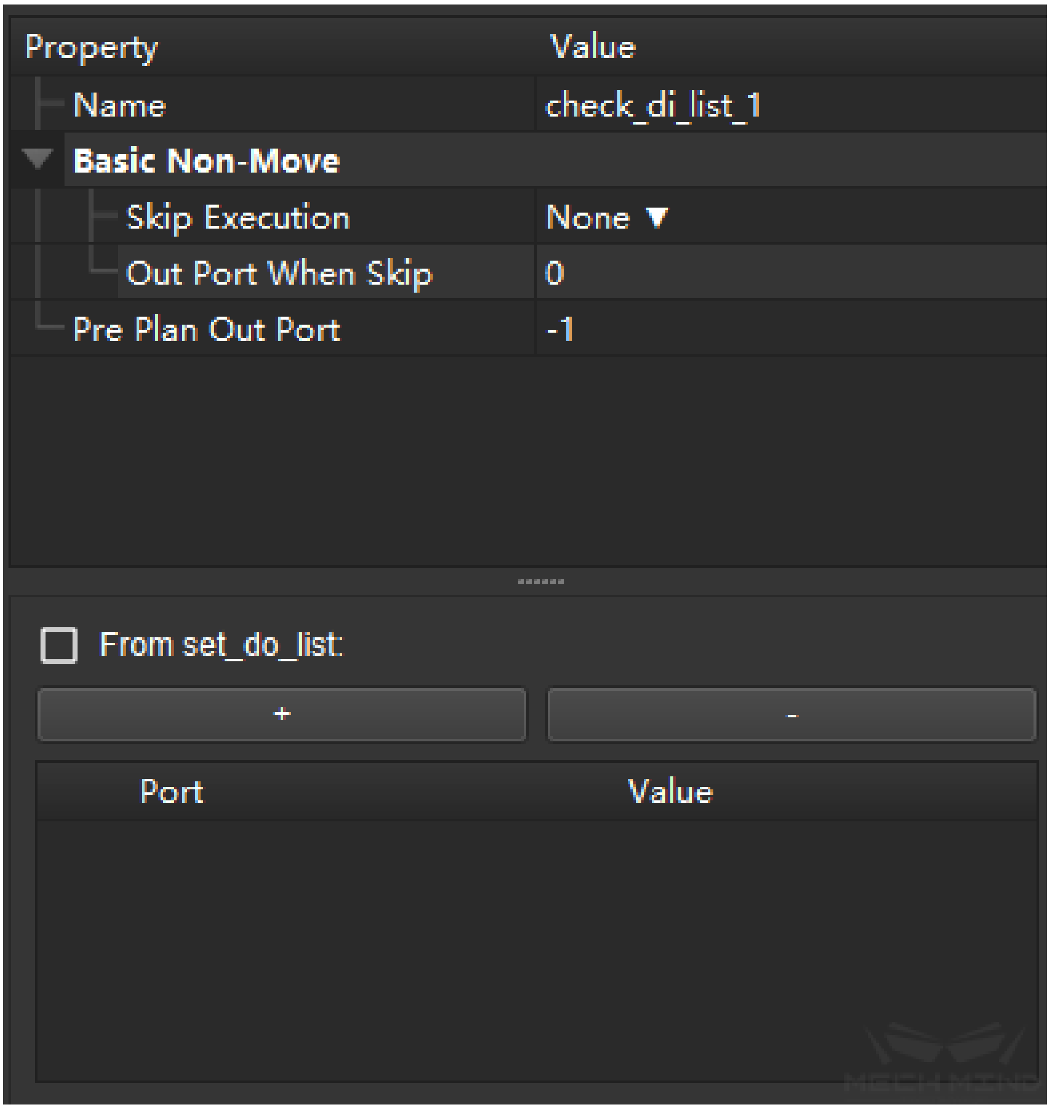
port DI port of the robot to be checked

prePlanOutPort Pre-planned output port

> Setting basis: the default value is -1. If the value is -1, the plan will be interrupted, that is, two relative tasks before and after this task cannot be planed together
>
> If the plan shall not be interrupted, write 0 when the task output from left port in most cases, otherwise write 1.

4.3.2 check_di_list

Description



Property	Value
Name	check_di_list_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When Skip	0
Pre Plan Out Port	-1

☐ From set_do_list:

Check if the signal values of multiple DI ports of the robot are same as the setting values. If all values are as expected, this task will output from left port, otherwise from right port.

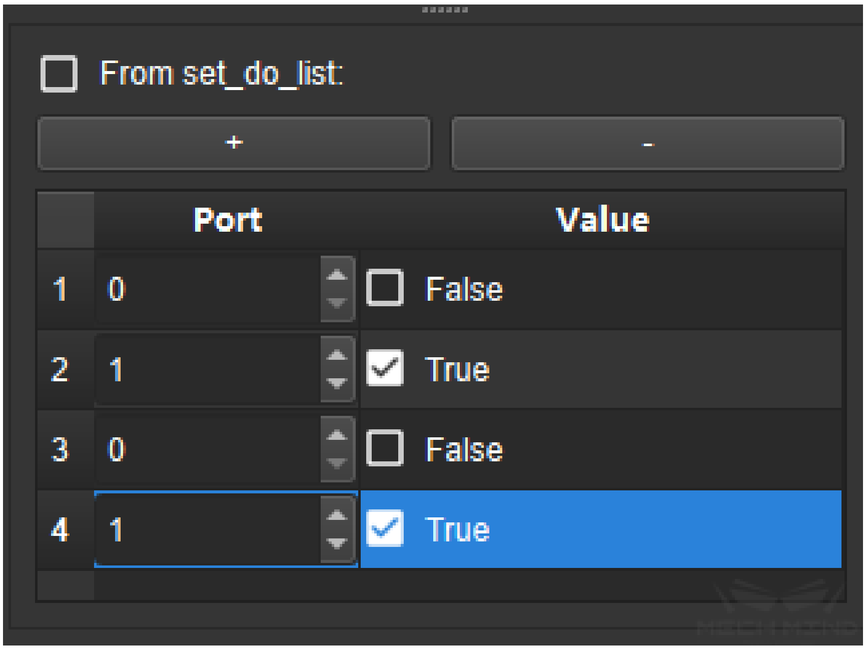
Parameters

BasicNonMove See *Other General Parameters*

ioMappingConfigPath Using with the *From Task* is checked

From Task

- If it is unchecked, add the port and corresponding value to be checked



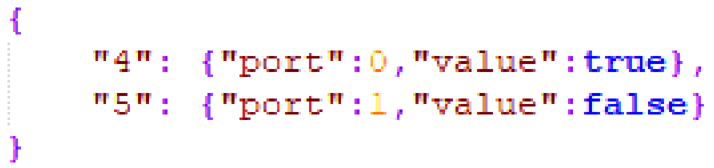
☐ From set_do_list:

	Port	Value
1	0	☐ False
2	1	☑ True
3	0	☐ False
4	1	☑ True

- If it is checked, select *set_do_list* which need to be checked and the json file corresponding to *ioMappingConfigPath*

  Application: Check if all the DO port are enabled as expected when using *set_do_list* control multiple suction cups.

  json file format:

  ```
  {
          "4":  {"port":0,"value":true},
          "5":  {"port":1,"value":false}
  }
  ```

  Note:

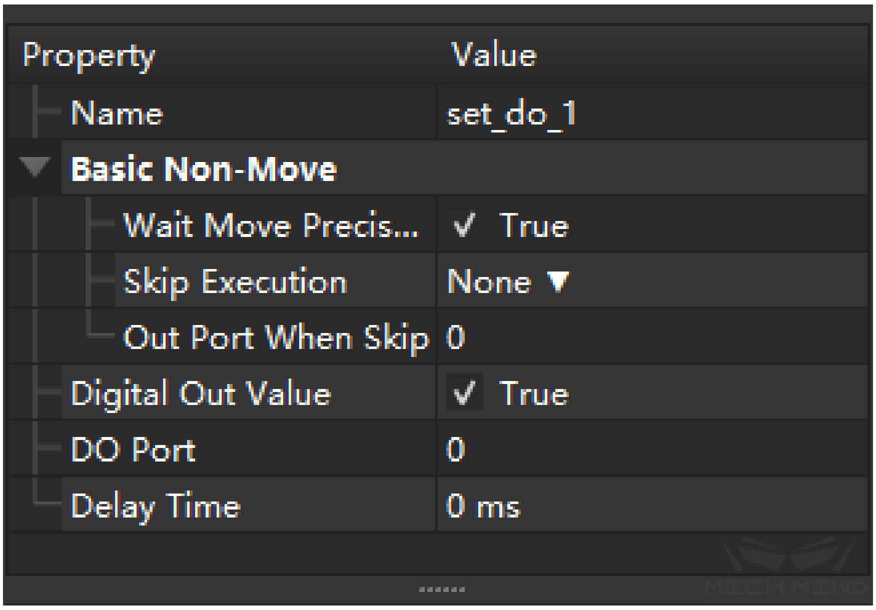
  By checking the port value of the selected *set_do_list* , find the real corresponding port to be checked in the json file and check whether the value is correct.

  As shown in the picture, 4, 5 are the ports set for the selected *set_do_list* , Port 4 is mapped to DI port 0, and port 5 is mapped to DI port 1. Then check if the value of port 0 is true and the value of port 1 is false.

  If a port in the selected *set_do_list* is not correspondingly found in the json file, an exception occurs.

4.3.3 set_do

Description



Property	Value
Name	set_do_1
Basic Non-Move	
Wait Move Precis...	✓ True
Skip Execution	None ▼
Out Port When Skip	0
Digital Out Value	✓ True
DO Port	0
Delay Time	0 ms

Set the specified DO port signal of the robot

Parameters

BasicNonMove See *Other General Parameters*

digital_out_value Set value of the robot digital output port, if it is checked as True, the value is 1, otherwise the value is 0.

port The digital output port number of Robot

delayTime Change the DO value after the setting time

4.3.4 set_do_list

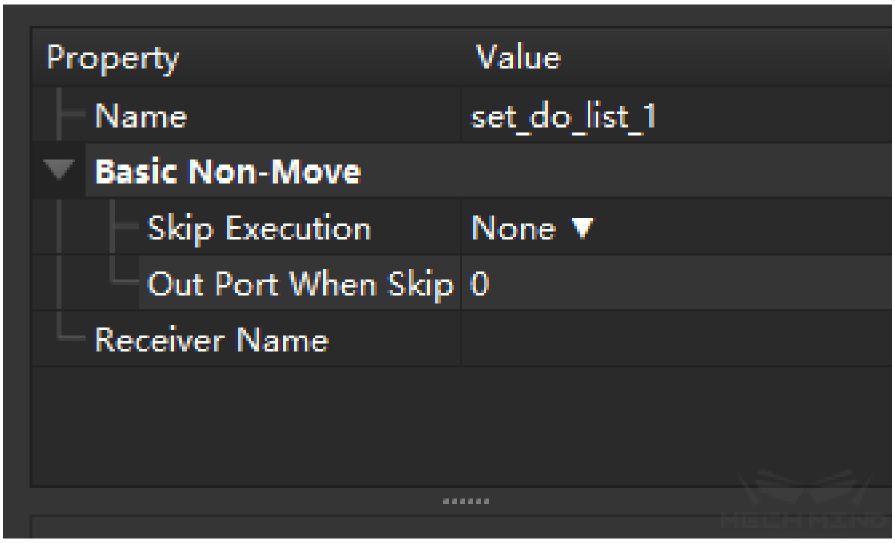
Description

Set the multiple specified DO ports signal of the robot

This module is used to control multiple digital outputs at the same time. There are two application scenes:

- Constantly enable control several suckers at the same time, the port number can be defined by the user directly on the interface
- Use individually controllable combined suckers, which require a sucker offset when grasping, or when grasping multiple boxes at one time, the *visual_move* provides the port number that needs to be controlled for opening

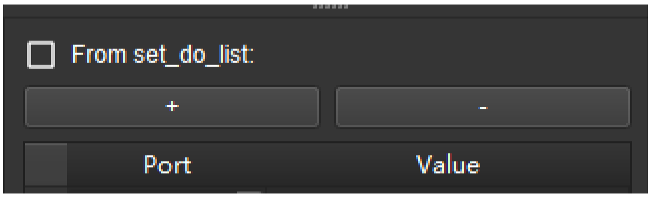
Parameters



Property	Value
Name	set_do_list_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When Skip	0
Receiver Name	

BasicNonMove See *Other General Parameters*

receiverName if the adapter is required to send the set DO status to an external device, enter the service name registered by the adapter on the Mech-Center to receive the IO status

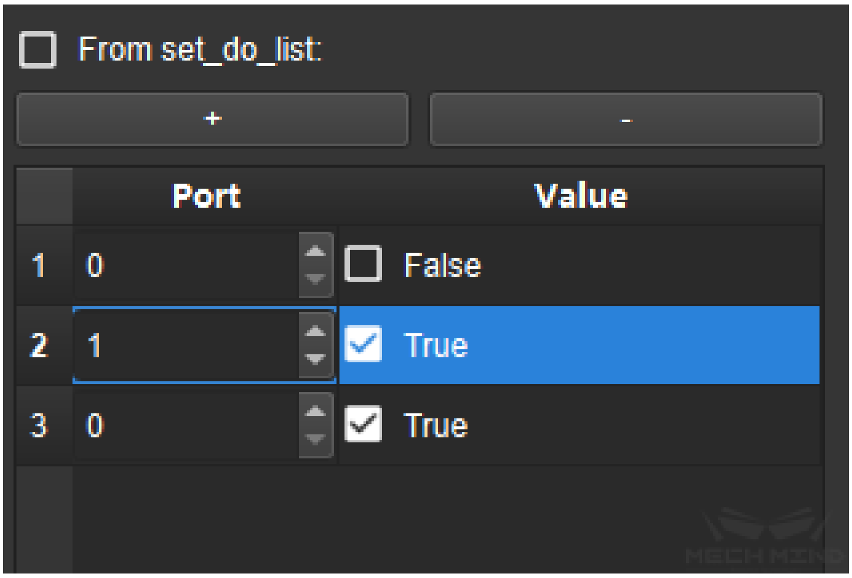


□ From set_do_list:

Port	Value

From Task:

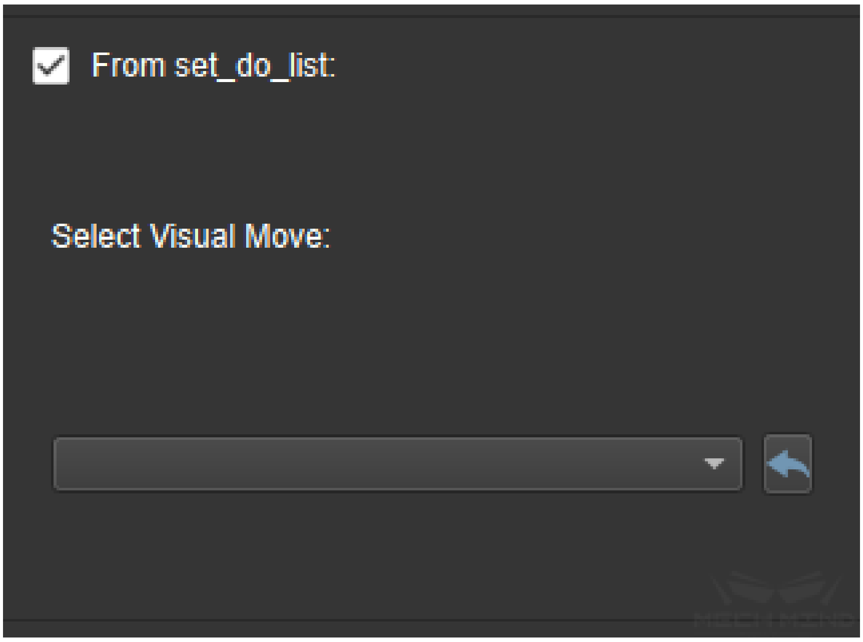
- When the DO port to be opened at the same time is fixed, it is unnecessary to check From Task. Add or delete the number of DO by + or - , and then change the corresponding port number and value, as shown in the figure below



☐ From set_do_list:

	Port	Value
1	0	☐ False
2	1	☑ True
3	0	☑ True

- When using *visual_move* and using the offset strategy or multipick strategy, the DO port to be opened each time is not fixed, which is provided by *visual_move* . At this time, check *From Task* , as shown in the figure below, and elect the corresponding visual_move name below.

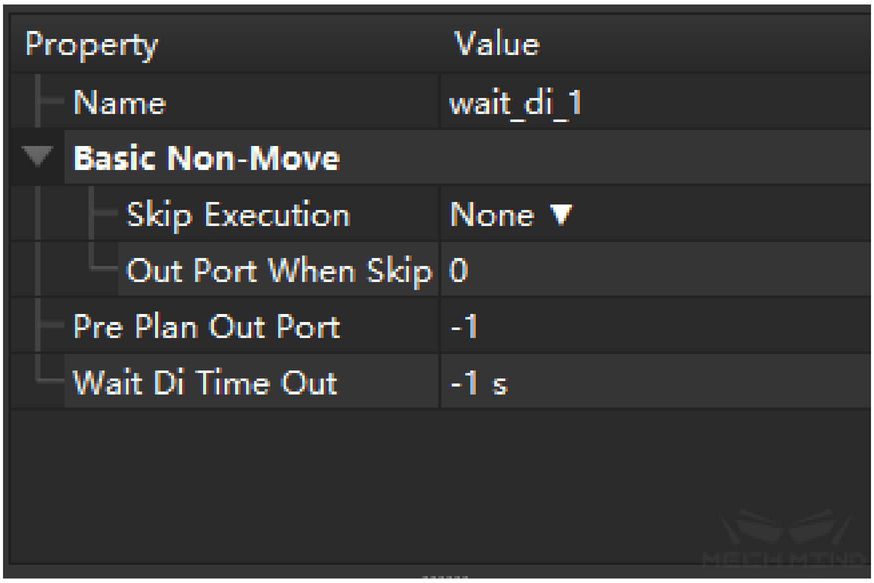


☑ From set_do_list:

Select Visual Move:

4.3.5 wait_di

Description



Property	Value
Name	wait_di_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When Skip	0
Pre Plan Out Port	-1
Wait Di Time Out	-1 s

Wait for the signal from the specified DI port of robot to reach the preset value

Parameters

BasicNonMove See *Other General Parameters*

waitdi_timeout Setting of pending for DI timeout. The default value is -1. When it is -1, it will keep pending.

prePlanOutPort Pre-planned output port.

Setting basis: the default value is -1. If the value is -1, the plan will be interrupted, that is, two relative tasks before and after this task cannot be planed together

If the plan shall not be interrupted, write 0 when the task output from left port in most cases, otherwise write 1.

4.4 Robot Related Tools

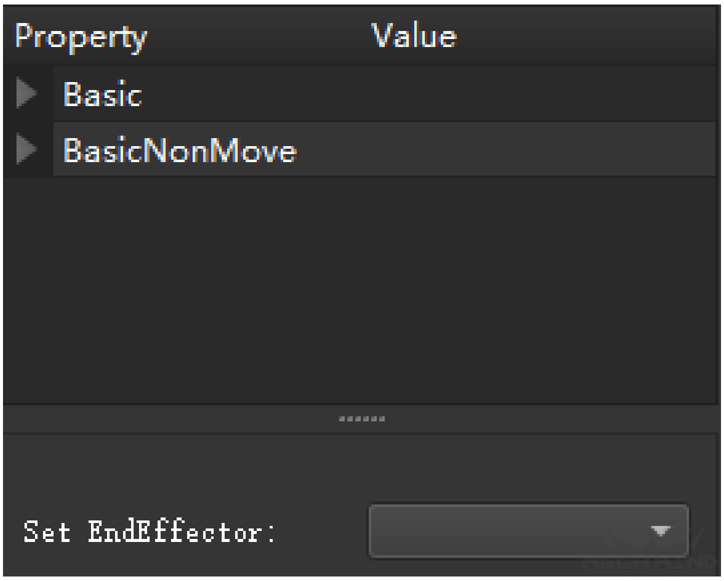
4.4.1 call_robot

4.4.2 check_tcp

Description

This module is applied to the scenario that robot switches end effectors to picking. It can check if the current end effector is the selected end effector, if same, it will output from left port, else it will output from right port.

Parameters



Set EndEffector:

BasicNonMove See *Other General Parameters*

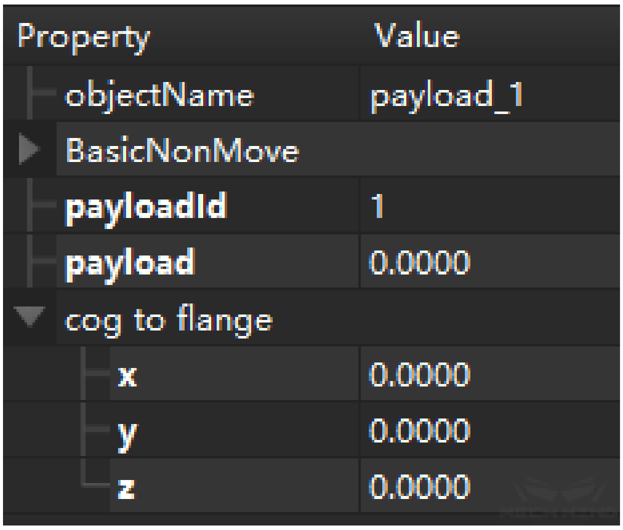
Set EndEffector Select the expected end effector to check if the current end effector is right.

4.4.3 payload

Description

This module is used for setting the payload of robot

Parameters



Property	Value
objectName	payload_1
BasicNonMove	
payloadId	1
payload	0.0000
cog to flange	
x	0.0000
y	0.0000
z	0.0000

BasicNonMove See *Other General Parameters*

payloadId If there exists multiple payloads, the index of payload in use need to be specified.

payload Fill the maximum payload including end effector and objects(kg)

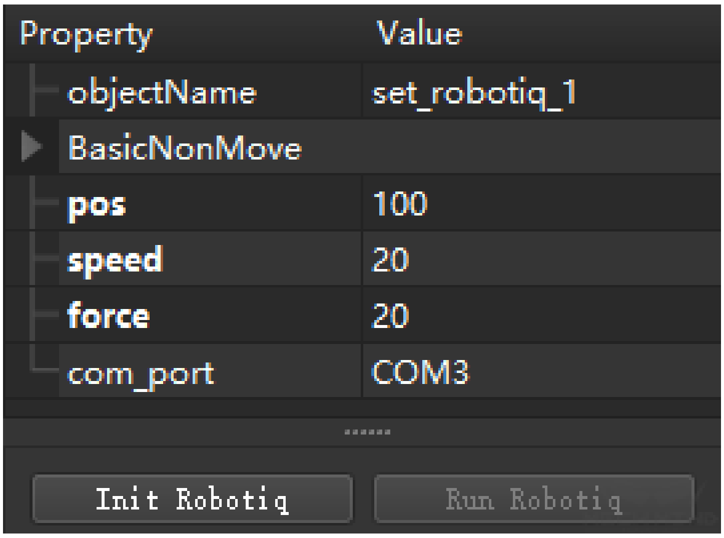
cog to flange x/y/z: Set the center of gravity of load in the flange coordinate system.(According to the brand of robot, not required)

4.4.4 set_robotiq

Description

This module enables the RobotIQ gripper to be controlled directly through Mech-Viz. RobotIQ gripper is controlled by serial port based on modbus protocol, PC or IPC can control its position/speed/force through 485 serial port (or USB to 485)

Parameters



Property	Value
objectName	set_robotiq_1
BasicNonMove	
pos	100
speed	20
force	20
com_port	COM3

BasicNonMove See *Other General Parameters*

pos Range: 0~255; 0 is the state where the jaws are opened the most

speed Action speed, range: 0~255.

force Torque, range: 0~255.

comPort The cluster communication port number connected to RobotIQ, it can be viewed in the device manager

Init Robotiq Initialize the COM connection, it will be initialized automatically during running and simulation

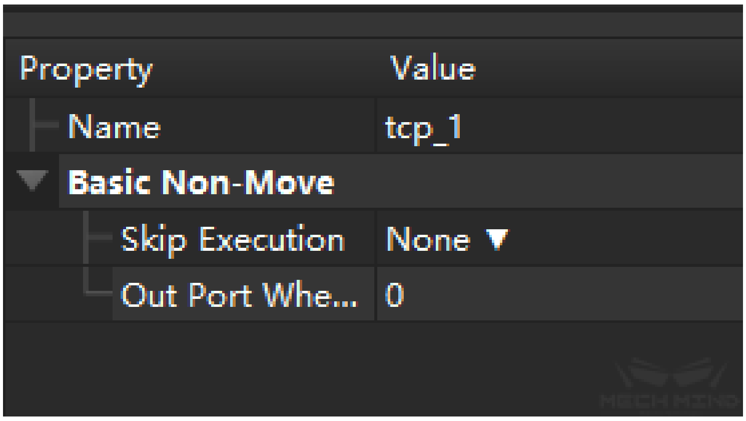
Run Robotiq After connecting with RobotIq, you can click this button, it will move the gripper according to the set parameters to preview

4.4.5 tcp

Description

Used in scenarios where different TCPs need to be switched; connect this module before grabbing, select the required end effector, and switch between different TCPs for grabbing.

Parameters



Property	Value
Name	tcp_1
Basic Non-Move	
Skip Execution	None ▼
Out Port Whe...	0

BasicNonMove See *Other General Parameters*

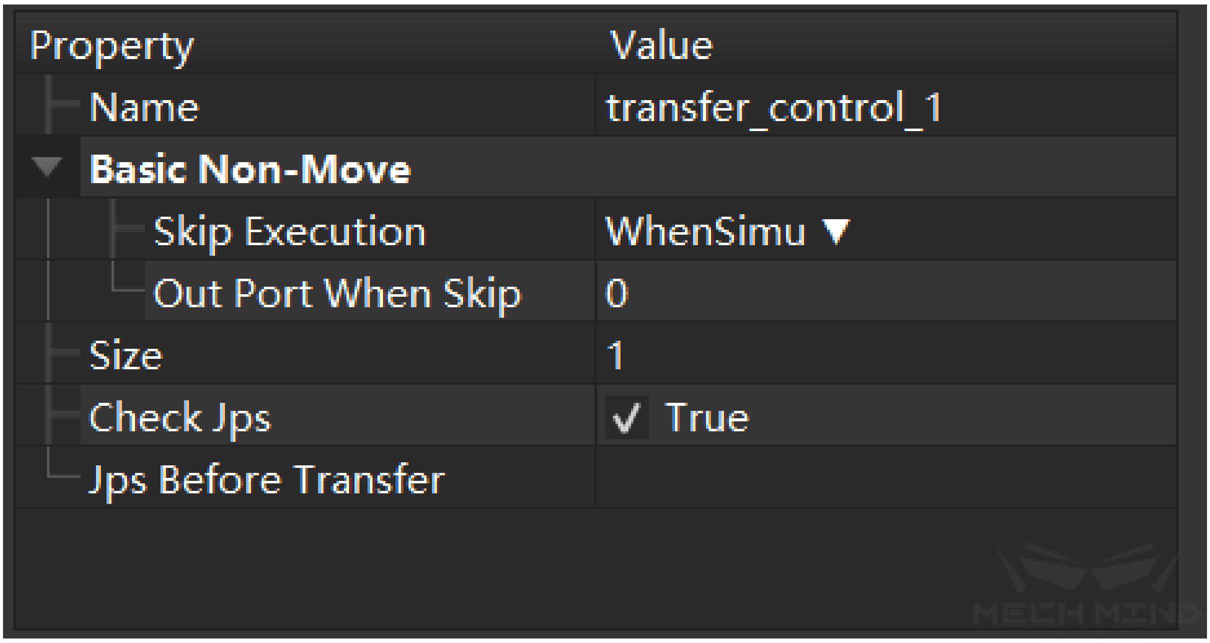
Set EndEffector In the drop-down menu is the end effector that has been added in *Robot Tab*, select the end effector to be used currently.

4.4.6 transfer_control

Description

This Skill is used to temporarily transfer control to the robot and wait to continue to control the robot.

Parameters



Property	Value
Name	transfer_control_1
▾ Basic Non-Move	
Skip Execution	WhenSimu ▼
Out Port When Skip	0
Size	1
Check Jps	✓ True
Jps Before Transfer	

BasicNonMove See *Other General Parameters* for details.

Size The number of output ports.

Check Jps Whether to check the joint angle of the robot before transferring control.

Jps Before Transfer This option will only appear when the *Check Jps* option is checked. You need to enter the Jps value required to check the Jps.

4.5 Service

Service module is mainly used for data interaction between Mech-Viz and external devices, and is generally used with adapters.

Note: It is required for such modules to be named in the properties, and notify the adapter developer.

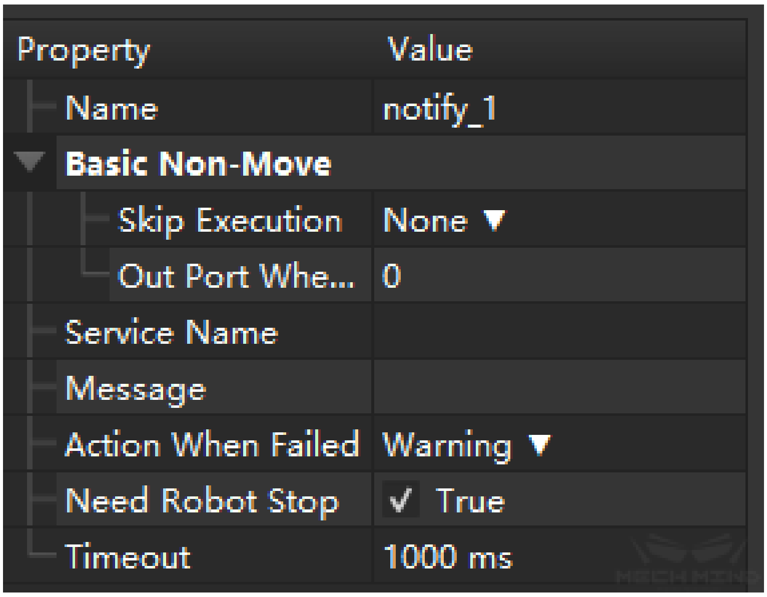
4.5.1 notify

Description

Use this module when the program proceeds to a node and the external device is required for the process

For example, when a certain branch is proceeded by the program and an external device is required to send a branch command again, a notify module can be connected at the end of this branch, and a message is sent to the adapter to obtain the command of the external device

Parameters



Property	Value
Name	notify_1
Basic Non-Move	
Skip Execution	None ▼
Out Port Whe...	0
Service Name	
Message	
Action When Failed	Warning ▼
Need Robot Stop	✓ True
Timeout	1000 ms

BasicNonMove See *Other General Parameters*

serviceName The adapter obtains the message of this module by service name. If there are multiple notify modules in the project, all notify modules should use the same service name

message The adapter executes different logic according to the content of the message, such as if the branch operation ends, the message can be set to: finish

actionWhenFailed when the notify message is not sent successfully, the actions done by the software

needRobotStop the default is True, the robot will pause when proceeding to this module; if set to False, the robot can send messages while running

timeout If no message is sent exceeding this time, perform the actions setted at actionWhenFailed, the unit is ms

..vision_proxy/vision_proxy

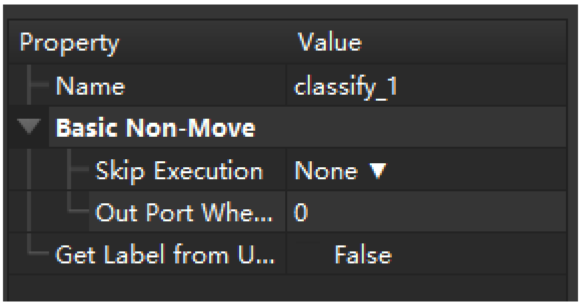
4.6 Tools

4.6.1 ▦ classify

Description

Run to different branches according to the object label corresponding to the actual grasp pose given by *visual_move*

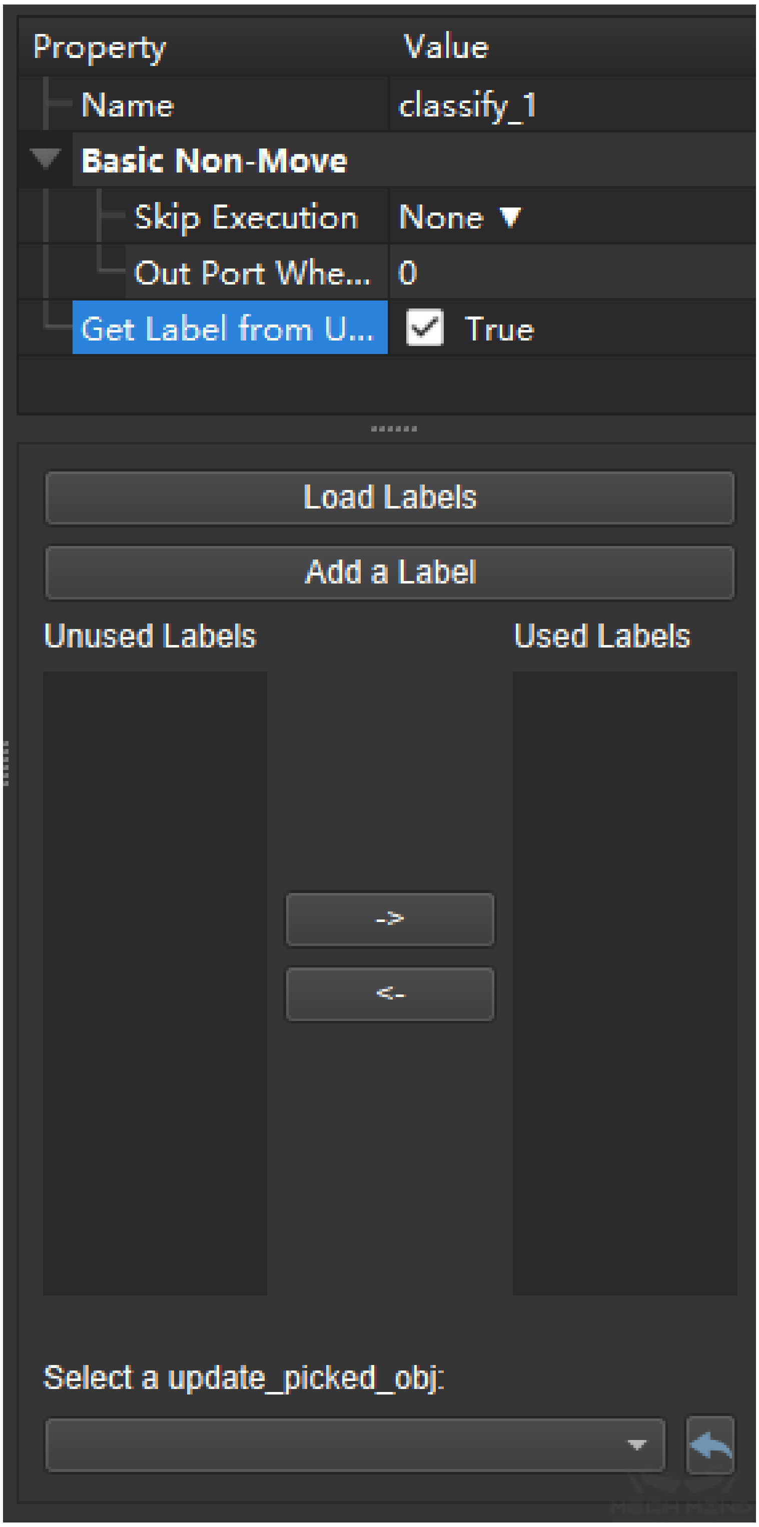
Parameters



Property	Value
Name	classify_1
Basic Non-Move	
Skip Execution	None ▼
Out Port Whe...	0
Get Label from U...	False

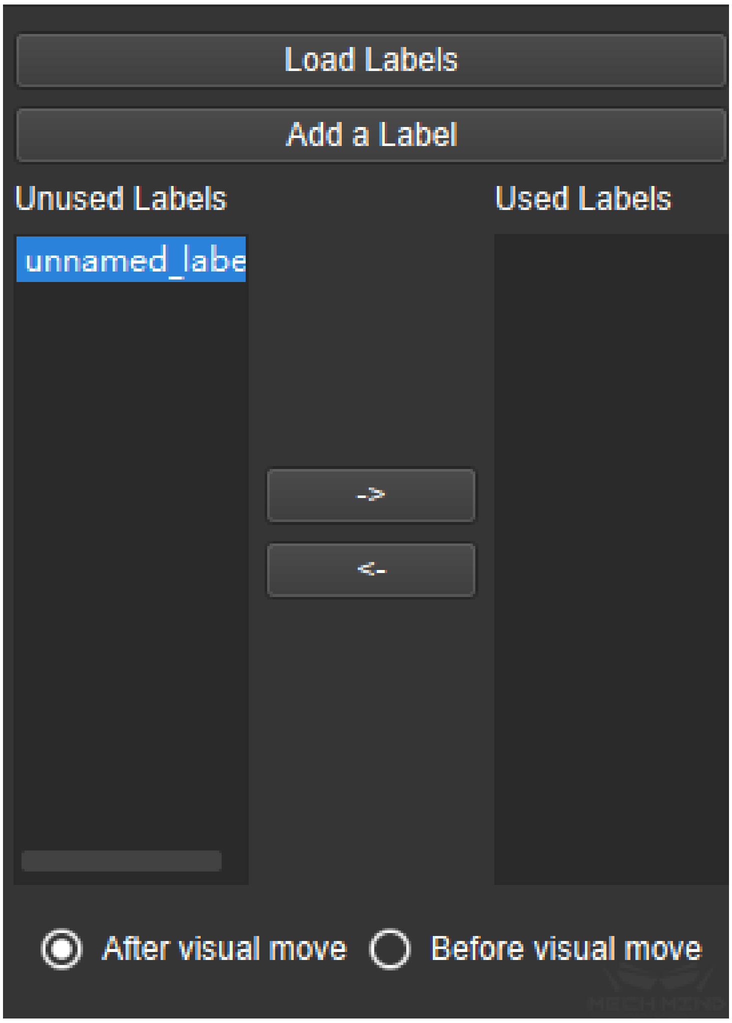
BasicNonMove See *Other General Parameters*

getLabelFromUpdatePickedObj Usage scenario: The label of grasp object can not be acquired from *visual_move* , it is acquired from second-time detection by *update_picked_obj* Once it is checked as True, the drop-down list *Select a update_picked_obj* will appear in the bottom, as shown in the picture, select the module name which need to be classified.



Property	Value
Name	classify_1
Basic Non-Move	
Skip Execution	None ▼
Out Port Whe...	0
Get Label from U...	☑ True

Load Labels Choose the labels.json file to load the predefined labels. Click –> or <– can add or delete corresponding port of labels



● After visual move ○ Before visual move

After visual move Default unselected, it can be choose when the objects need to be placed in different position according to labels after picking. According to the object label corresponding to the actual grasp pose given by *visual_move* , the program will run into different branch

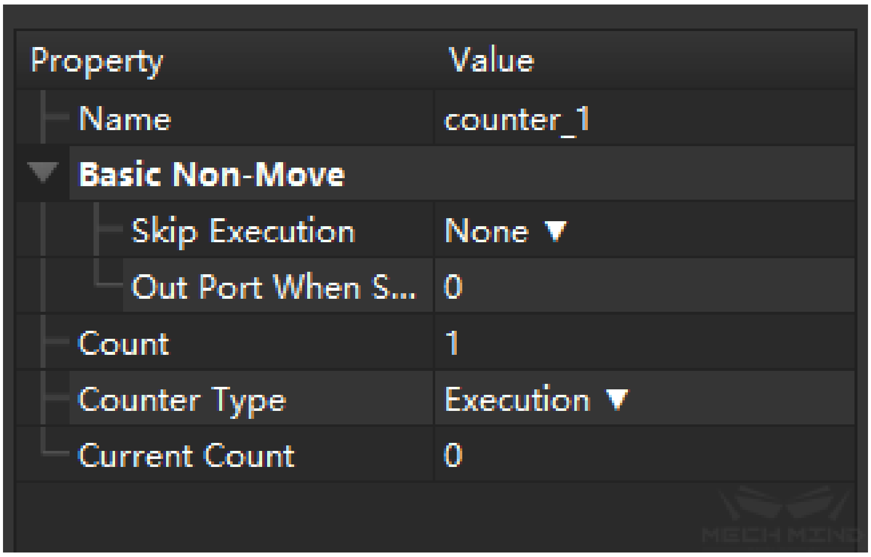
Before visual move If the grasping depends on the labels given by current detection so that using different end effectors, this function can be choose. Due to this function is relate to the actual picked object' s label given by subsequent *visual_move* , the dependent module need to be selected in the drop-down list *Select a visual_move*

4.6.2 counter

Description

Count the execution times of mission or *branch_by_service_message* , or count the picking times. Generally it is used with *reset_task* .

Parameters



Property	Value
Name	counter_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Count	1
Counter Type	Execution ▼
Current Count	0

BasicNonMove See *Other General Parameters*

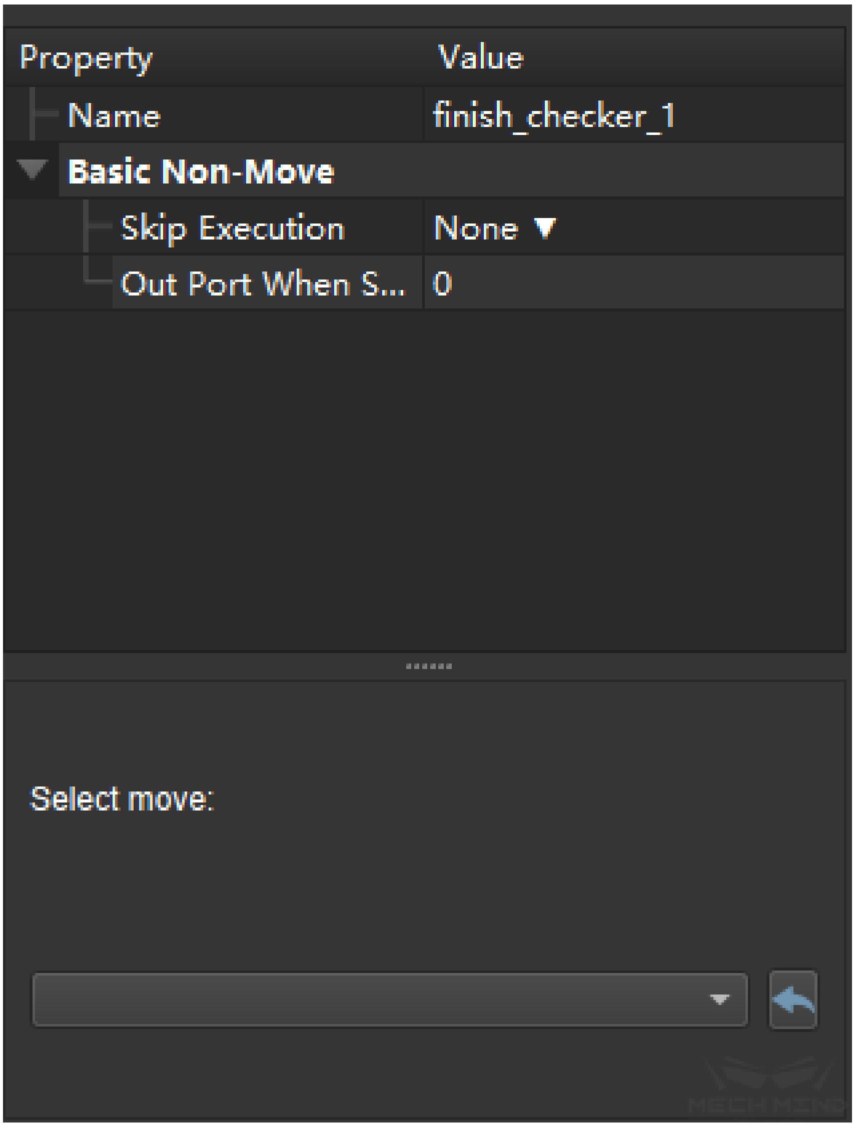
countertype Types of counts: execution and pickedcount

4.6.3 finish_checker

Description

It is used to ensure *Movement Tasks* with index such as *move_list* , *move_grid* ,etc. finish execution in a single cycle to avoid loops.

Parameters



Property	Value
Name	finish_checker_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0

Select move:

BasicNonMove See *Other General Parameters*

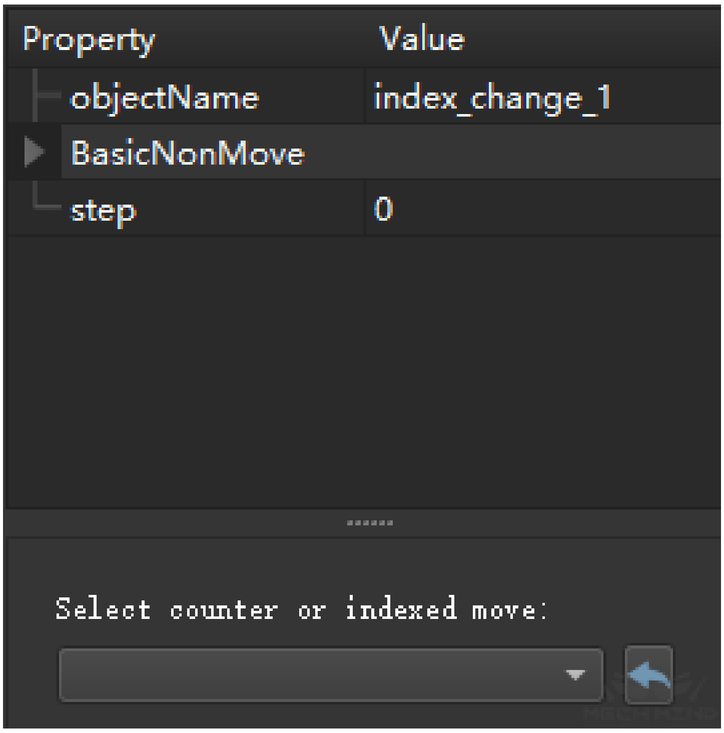
Select indexed move Select the movement module which need to be checked. Exit from port 1 if it is completed, exit from port 0 if it is not completed.

4.6.4 index_change

Description

This module can be used with all the *Movement Tasks* which has the property of *curIndex* , or *counter* . It will change the *curIndex* of specified movement task or the *count* of counter according to *step* every execution.

Parameters



Property	Value
objectName	index_change_1
▶ BasicNonMove	
step	0

Select counter or indexed move:

BasicNonMove See *Other General Parameters*

step The number that increase/decrease the current index or count.

For example: If the step is 0, it will not change the index/count of specified module, and the module will accumulate the execution times; While the step is -1, the index/count of specified module will minus 1 every execution; so on and so forth.

Select counter or indexed move Select the *counter* whose count need to be changed, or *Movement Tasks* whose index need to be changed.

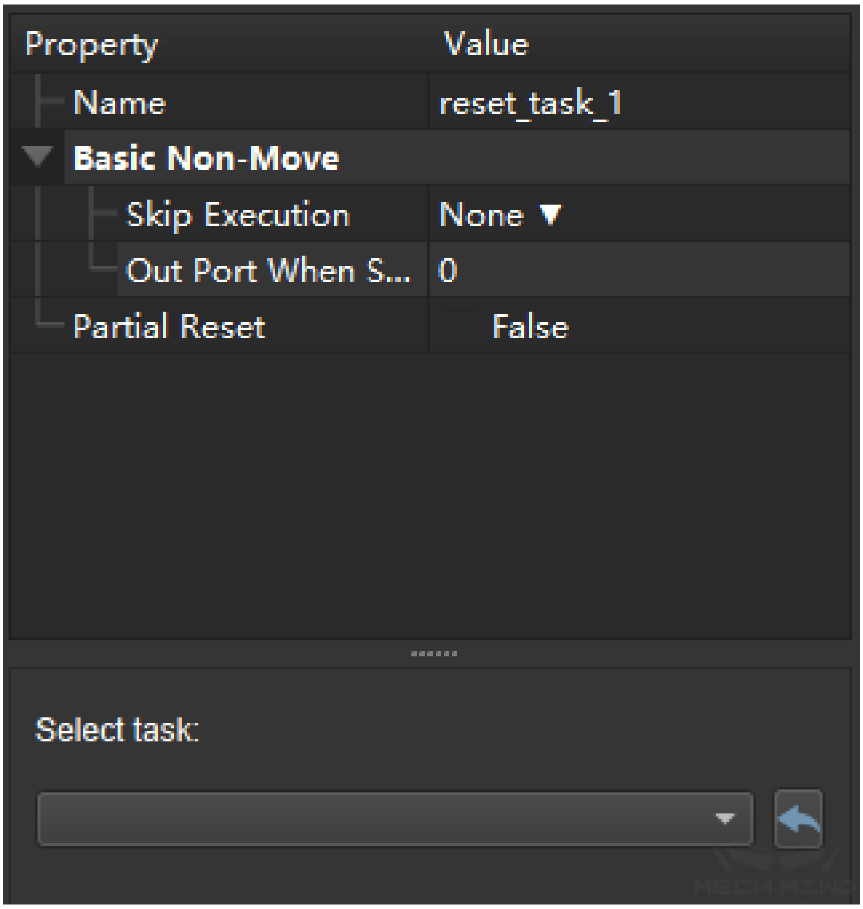
4.6.5 reset_task

Description

Reset the selected module, it can be used:

- To reset the indexed move or counter, such as *move_grid* , *counter*
- To reset the move task which move target is “Place” , it will clear the placed object in scene
- To reset the last *visual_move* when multiple *visual_move* share vision results, it will clear the vision result of this detection
- To trigger some special function. Such as Pallet task which *needAdjustPalletPose* is checked, except the first time running, it will also update the pose of pallet after being reset

Parameters



Property	Value
Name	reset_task_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Partial Reset	False

Select task:

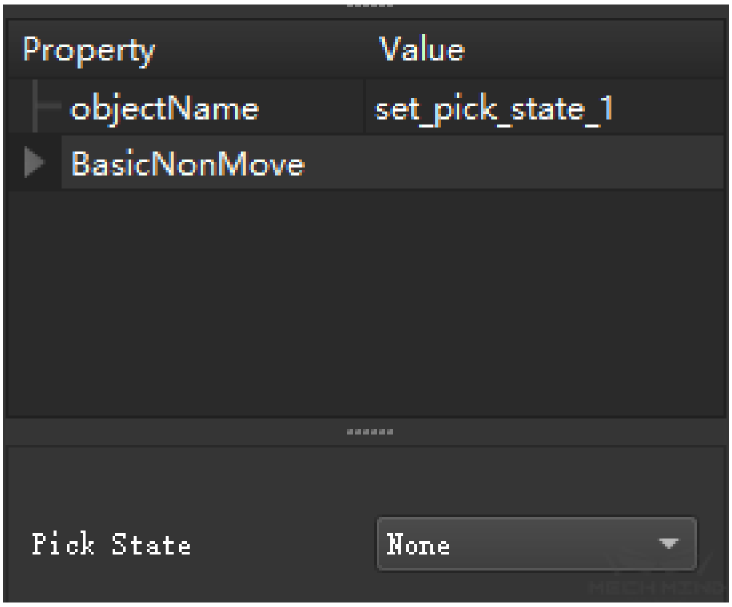
Select indexed move or counter Select the module name which need to be reset in the drop-down list

4.6.6 set_pick_state

Description

This module generally is used for changing the pick state if picking failed

Parameters



Property	Value
objectName	set_pick_state_1
BasicNonMove	

Pick State: None

BasicNonMove See *Other General Parameters*

Pick State Select the pick state to `Hold` or `None` according to the previous logic

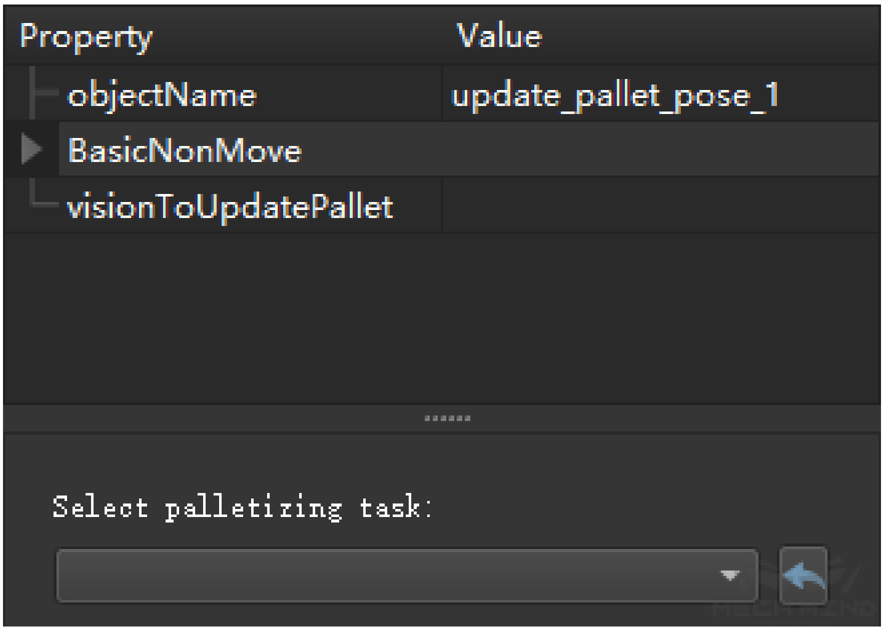
4.6.7 update_pallet_pose

Description

Update the pallet pose according to the vision result. The camera can be triggered at any time/robot pose to update the pallet pose.

If the pallet pose need to be updated before each palletizing, check *AdjustPalletViaVision* in the task of *Pallet*.

Parameters



Property	Value
objectName	update_pallet_pose_1
BasicNonMove	
visionToUpdatePallet	

Select palletizing task:

BasicNonMove See *Other General Parameters*

visionToUpdatePallet The name of the vision project which will update the pallet pose

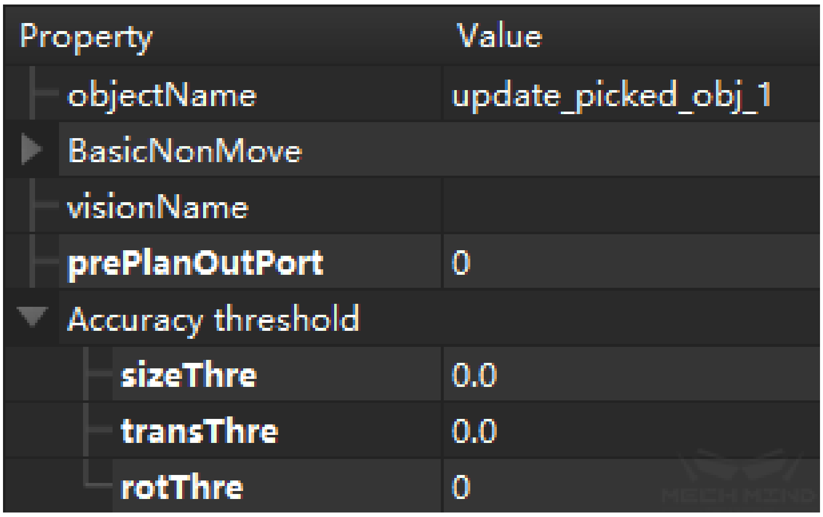
Select palletizing task Select the name of the palletizing module that needs to update the pallet pose in the project

4.6.8 update_picked_obj

Description

Update the size and pose of the grasped object by second-detection

Parameters



Property	Value
objectName	update_picked_obj_1
BasicNonMove	
visionName	
prePlanOutPort	0
Accuracy threshold	
sizeThre	0.0
transThre	0.0
rotThre	0

BasicNonMove See *Other General Parameters*

visionName The vision project name to second detect

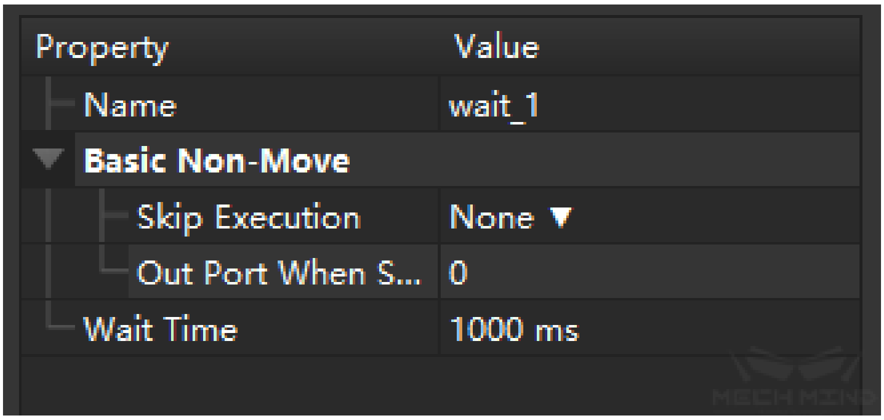
prePlanOutPort Default is 0, mudule will output from the left port “NoNeedUpdate” to speed up the project. It can be set to -1 if it is not neccessary to preplan.

Accuracy threshold sizeThre/transThre/rotThre: if the size/translations/rotation of box are all smaller than the specified threshold, it will be considered as the same, this module will output from the left port “NoNeedUpdate”

4.6.9 wait

Description

When execute this module, wait for the specified time, unit: ms



Property	Value
Name	wait_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Wait Time	1000 ms

Parameters

BasicNonMove See *Other General Parameters*

wait_time The duration to wait, the unit is ms

4.7 Topology

Icon	Module name	Function
	mission	*Multi-Level Mission Processing*
	set_tag	Identify the current branch route
	branch_by_tag	When the running of the common module is completed, Split into the original branch based on the ID and continue to run. It is used with a SetTag module.
	branch_by_service_message	*branch_by_service_message*
	mission_exit	Exit mission from specified exit port, it is used in mission

4.7.1 branch_by_service_message

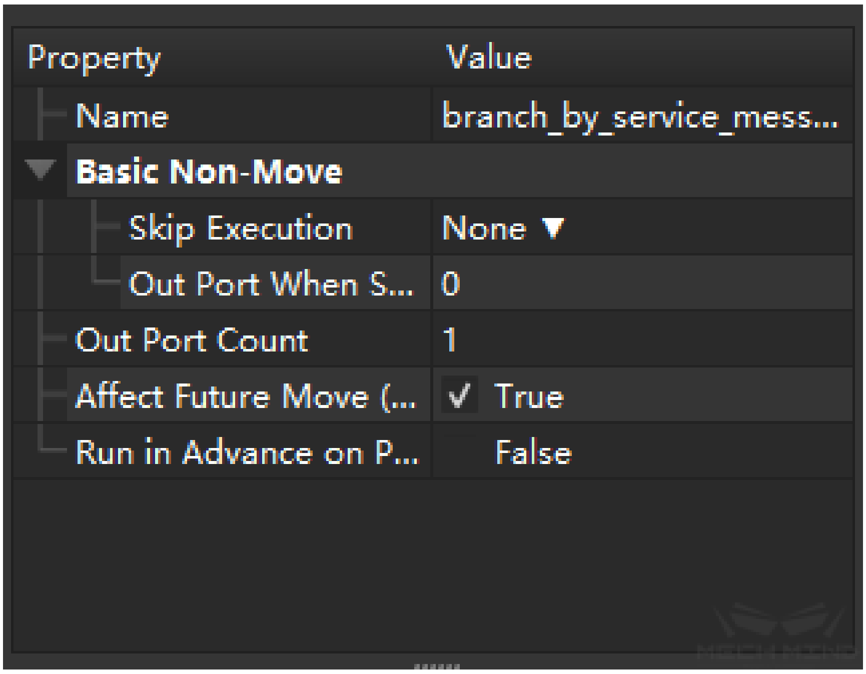
Description

When an external device sends a branch command to the adapter, the adapter calls this module after processing the command, and runs the corresponding branch

This module performs two functions in the program:

- Run the corresponding branch process based on the data of the external device
- Interrupt the running program, and continue to run when the external signal is received

Parameters



Property	Value
Name	branch_by_service_mess...
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Out Port Count	1
Affect Future Move (...	✓ True
Run in Advance on P...	False

objectName Since the adapter calls this module by names, it is required to change the object name when using this module

- When running different branches, the names can be written as: branch_1, branch_2
- If it is used for interruption, the names can be written as: interrupt_1, interrupt_2

BasicNonMove See *Other General Parameters*

size The number of output ports for this module, for example, the size is 3, this module has 3 output ports, which can be connected to three processes respectively; if this module is used for interruption, set size to 1.

affectFutureMovement True by default, interrupting the planning of subsequent logic in the software (when this module is used to execute different branches, since it does not know which branch will be run in the future, it will definitely affect program planning);

If it is only used to interrupt and wait for external signals, the subsequent program logic will not change. This parameter can be set to False.

runInAdvanceOnPlanned The default is False, the module will not receive commands from external devices in advance; if True is checked, the commands sent by external devices can be temporarily stored. When the program is proceeded to this module again, the temporary stored branch number will run directly to optimize the process.

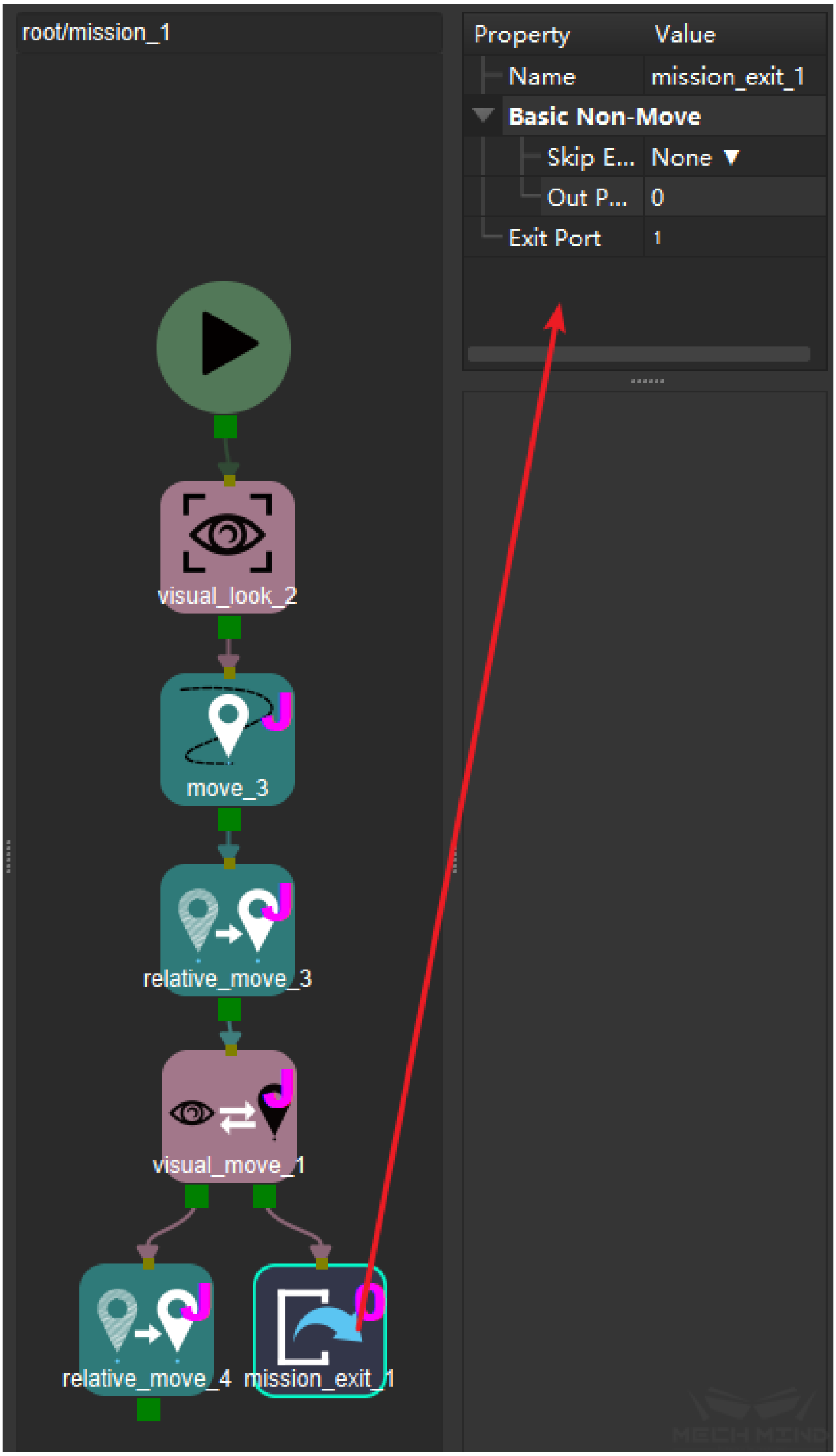
4.7.2 branch_by_tag

4.7.3 mission & mission_exit

mission is used to process multi-layer tasks.

mission_exit is applied in the situation that there are multiple possible results by the program, select an exit port to exit current mission; if the multiple outputs are not involved in mission, use mission alone.

Example of mission_exit is used



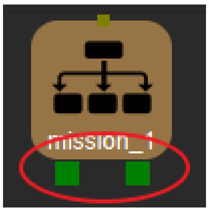
root/mission_1

Property	Value
Name	mission_exit_1
Basic Non-Move	
Skip E...	None ▼
Out P...	0
Exit Port	1

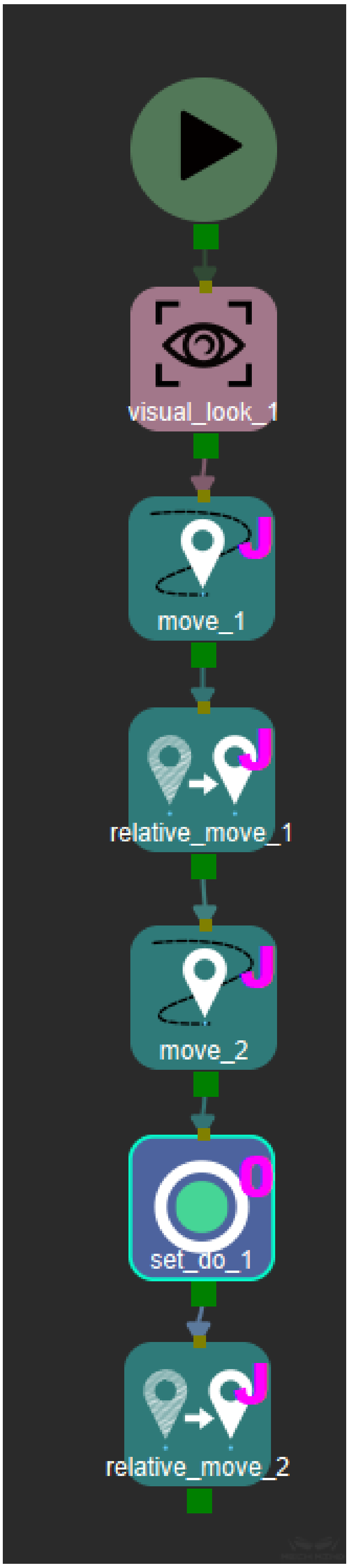
As shown in the figure, there is a visual_move in this mission. Due to the possibilities of visual recognition: success and no pose or failed to pick, the two exits are required to connect different processing logics respectively. Perform the following operation:

- Connect mission_exit module at the port2 of visual_move;
- Set the exit_port in the property of mission_exit to 1. (exit_port defaults to 0)

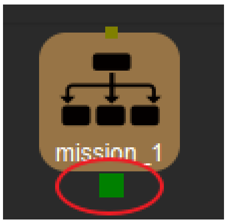
Back to the previous layer, you may see that the mission becomes two output ports, and users can connect them with different processing logic as required.



Example for using mission alone



In the figure above, the process flow in mission does not have multiple possible results. This mission has only one output port. As shown in the figure below, when the program finishes running the last module, it will jump out of mission directly.

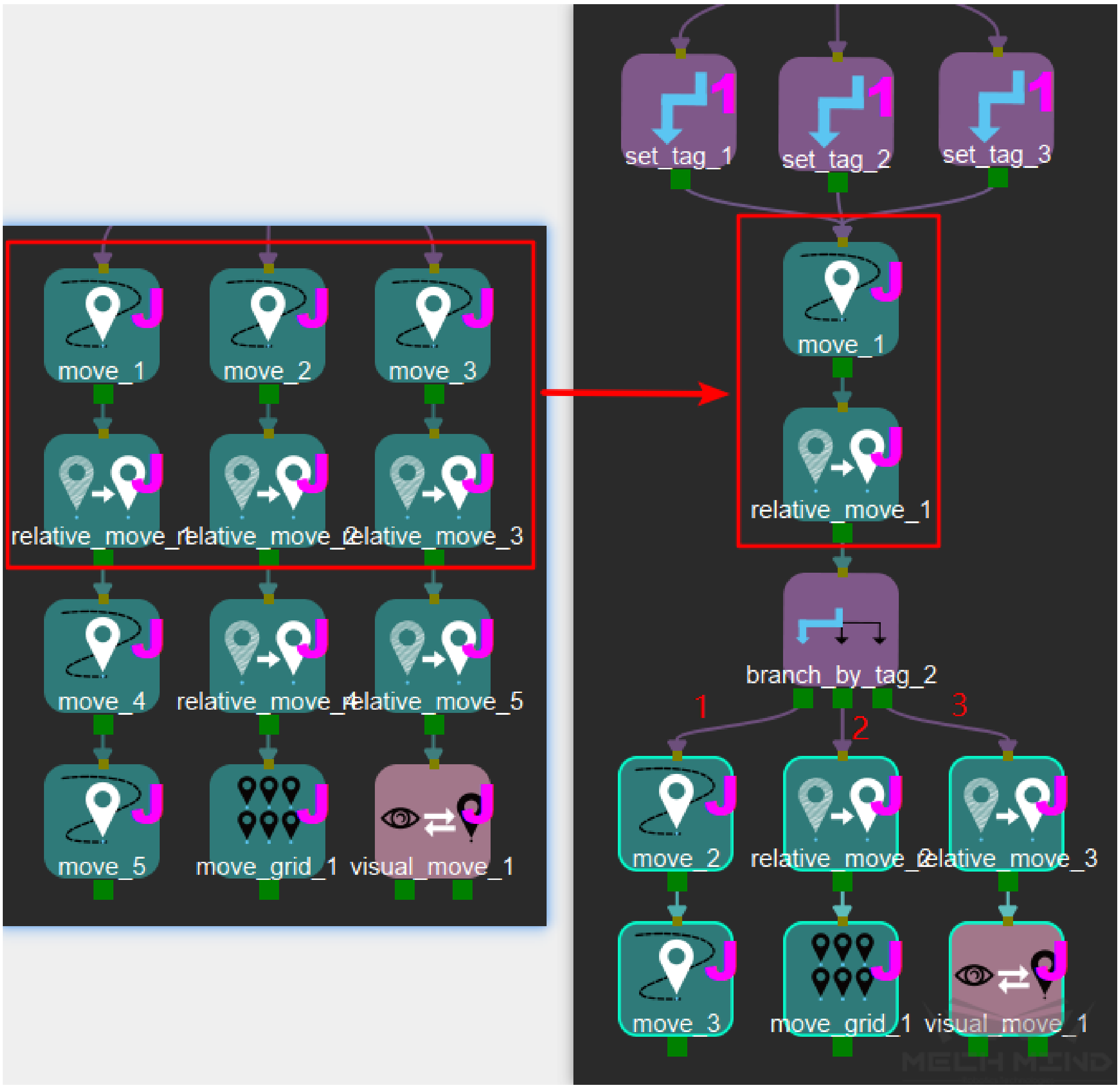


4.7.4 mission_exit

4.7.5 set_tag & branch_by_tag

The scene for this combination is: when there is a common part in multiple logical branches, the common part is not required to be copied by multiple times, use set_tag to mark the current branch, merge it into the common part, and then use branch_by_tag module to restore the program running logic to the original branch.

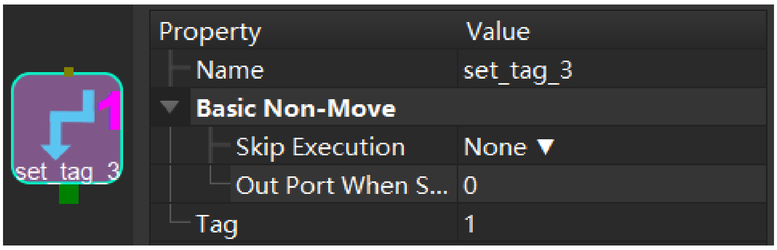
The following figure is an example:



The three branches have common modules: move and relative_move. No matter which branch, they will proceed to the same position when completing the previous process. At this time, the same module can be selected, and set_tag modules are added to the branches to mark the current branch number. When the program finishes move and relative_move modules, branch_by_tag will restore the program running logic to the previously running branch according to the current identification.

Parameter settings

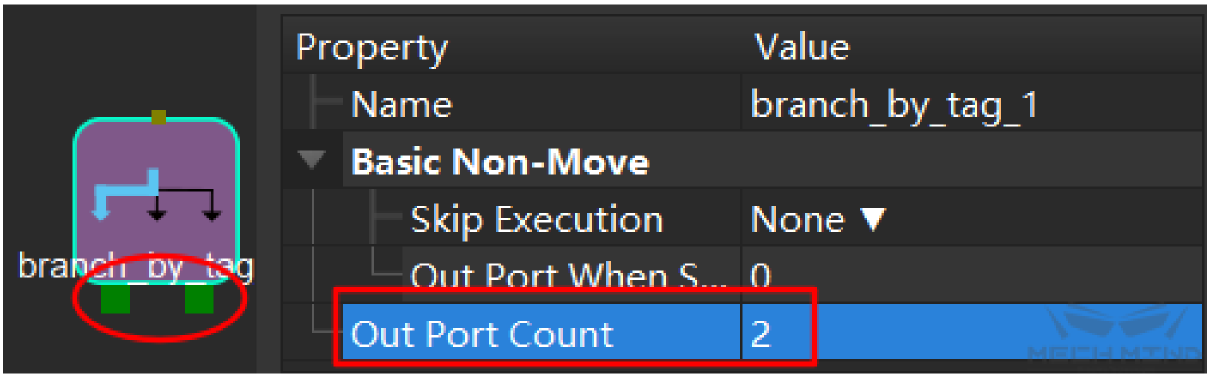
1. set_tag



Property	Value
Name	set_tag_3
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Tag	1

- Basic parameters for non-movement, see *Other General Parameters*
- routePort: the ID of the current branch, used to restore the branch of the branch_by_tag module

2. branch_by_tag



Property	Value
Name	branch_by_tag_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Out Port Count	2

- Basic parameters for non-movement, see *Other General Parameters*
- size: output ports of the branch_by_tag module, based on the total number of routePort

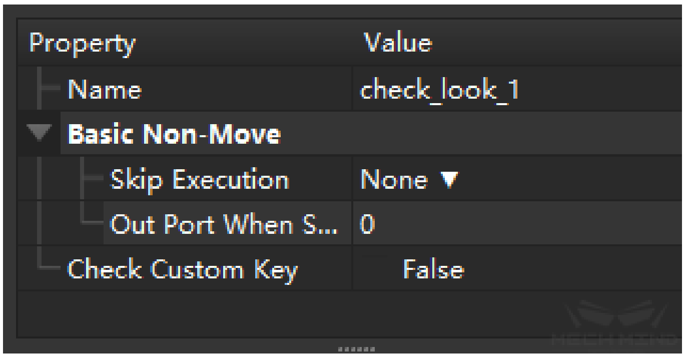
4.8 Vision

4.8.1 check_look

Description

Connect with *visual_look* to make project do different processes based on the vision result.

Parameters



Property	Value
Name	check_look_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Check Custom Key	False

BasicNonMove See *Other General Parameters*

Get Vision Services Select *visual_look* which need to be checked

Branch explanation

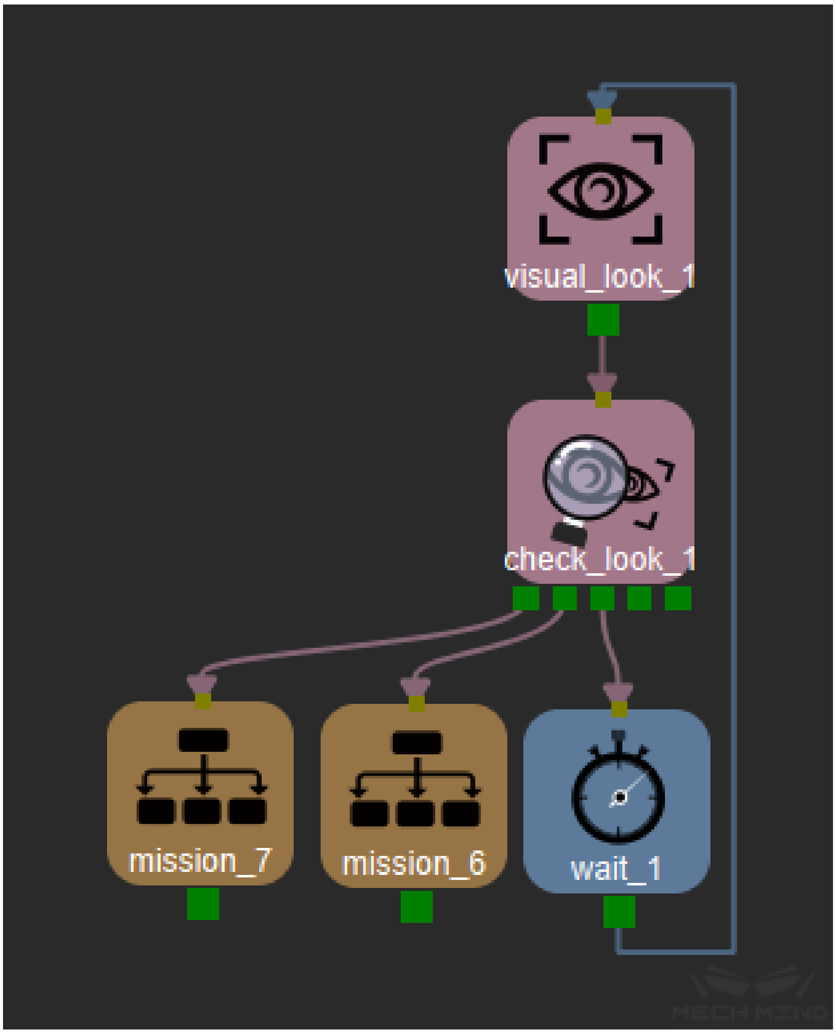
hasResult: If the visual recognition provides the result, then take this exit.

noResult: If the visual recognition does not provide the result, then take this exit.

notCalled: Applicable to multiple *visual_look*, used in loop nesting. This exit can be ignored if it follows *visual_look*.

unfinished: The visual processing is not finished. This exit usually follows waiting and returns to the entrance.

noCloudInRoi: No point cloud is in the Region of Interest (ROI) of vision (It is often used to determine whether there are any object and distinguish unidentified objects).



Output of check_look

4.8.2 set_pick_box

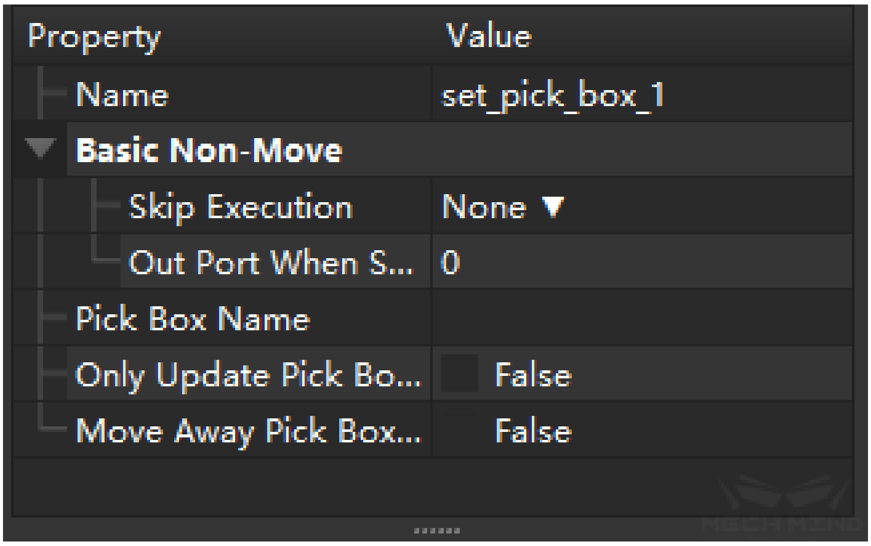
Description

Update the picking box pose and size according to the vision result.

It is usually used in the situation that the picking box pose or size need to be updated after grasping the whole objects in the picking box.

If the picking box pose and size need to be updated every detection, it is not necessary to use this module, fill the *pickBoxName* of *visual_move* directly.

Parameters



Property	Value
Name	set_pick_box_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0
Pick Box Name	
Only Update Pick Bo...	□ False
Move Away Pick Box...	False

BasicNonMove See *Other General Parameters*

pickBoxName Fill the picking box model name.

onlyUpdateBoxSize The default is true, it will only update the size of picking box; If it is checked as False, both size and pose will be updated.

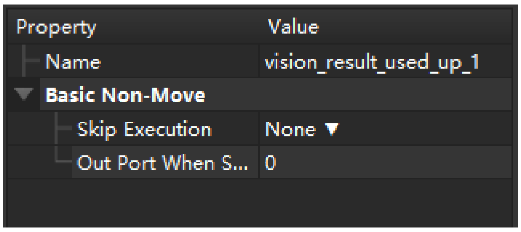
Vision Name Select the vision project name registered on Mech-Center; click *Get Vision Services* to refresh the registered services.

4.8.3 vision_result_used_up

Description

This module is used only if the *reuseVisionResult* of *visual_move* is checked. Check if all visual results in the current detection have been used up. This task is only for checking.

Parameters



Property	Value
Name	vision_result_used_up_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When S...	0

BasicNonMove See *Other General Parameters*

Select visual move: Select *visual_move* which need to be checked

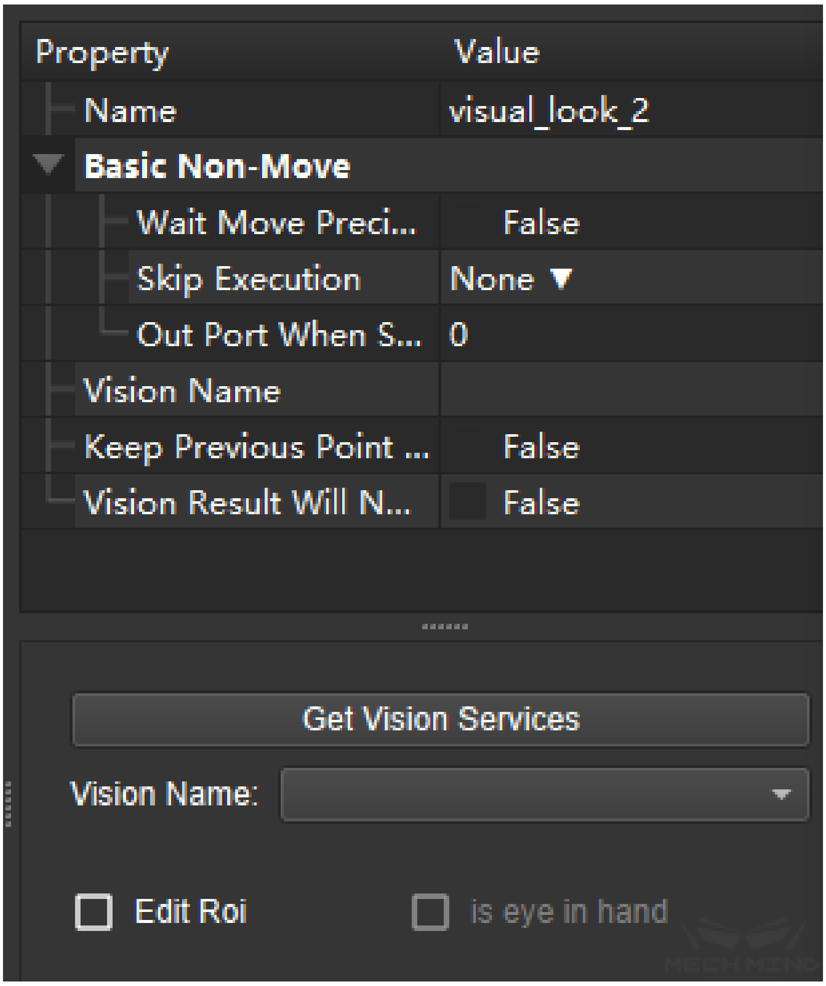
4.8.4 visual_look

Description

Call the vision service (run vision project) and transmit the result to the ROI in Mech-Viz

ROI: Region of Interest

Parameters



Property	Value
Name	visual_look_2
Basic Non-Move	
Wait Move Preci...	False
Skip Execution	None ▼
Out Port When S...	0
Vision Name	
Keep Previous Point ...	False
Vision Result Will N...	False

Vision Name:

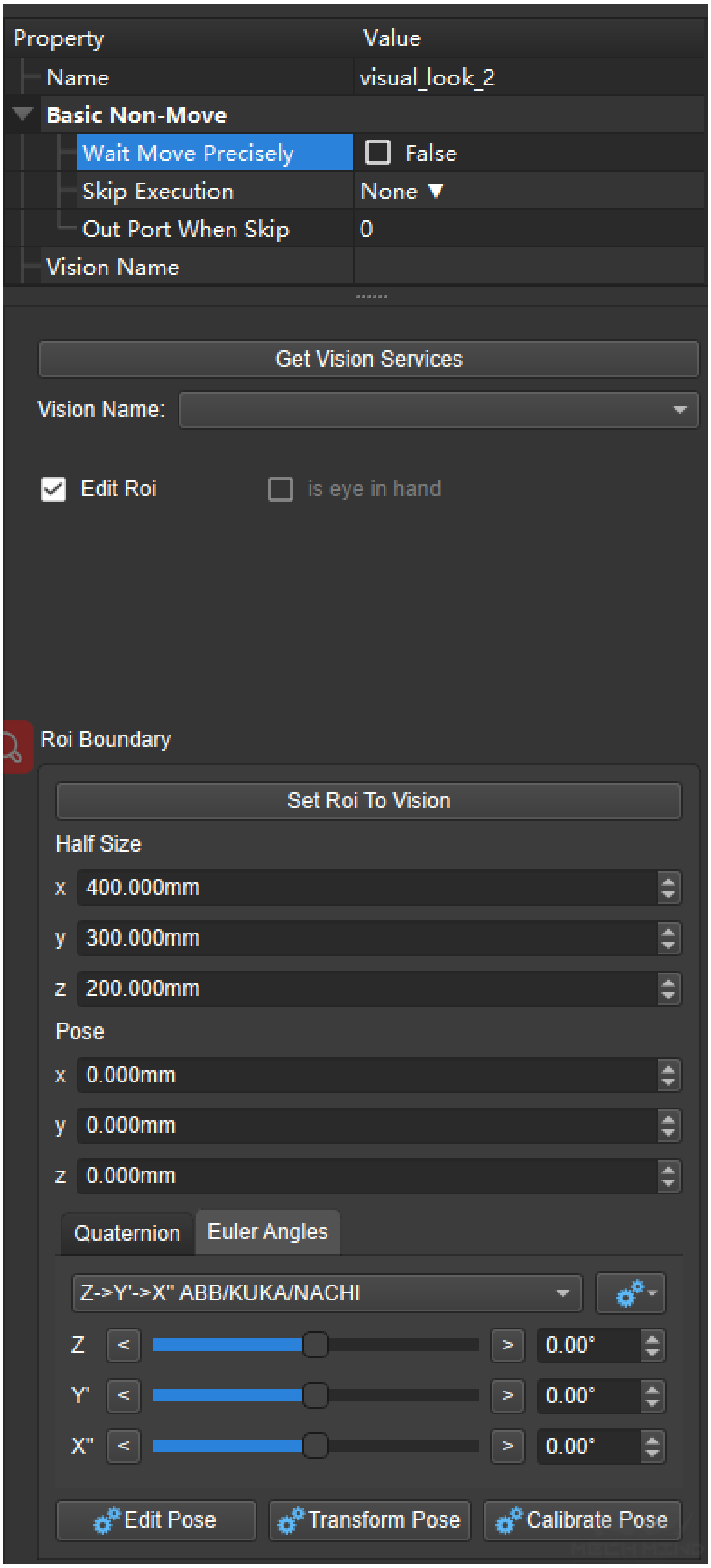
□ Edit Roi

□ is eye in hand

BasicNonMove See *Other General Parameters*

Vision name Select the vision project name registered on Mech-Center; click Get Vision Services to refresh the registered services.

Edit ROI If it is check, ROI Boundary can be directly edited in Mech-Viz.



Property	Value
Name	visual_look_2
Basic Non-Move	
Wait Move Precisely	□ False
Skip Execution	None ▼
Out Port When Skip	0
Vision Name	

Vision Name:

☑ Edit Roi □ is eye in hand

Roi Boundary

Half Size

x 400.000mm

y 300.000mm

z 200.000mm

Pose

x 0.000mm

y 0.000mm

z 0.000mm

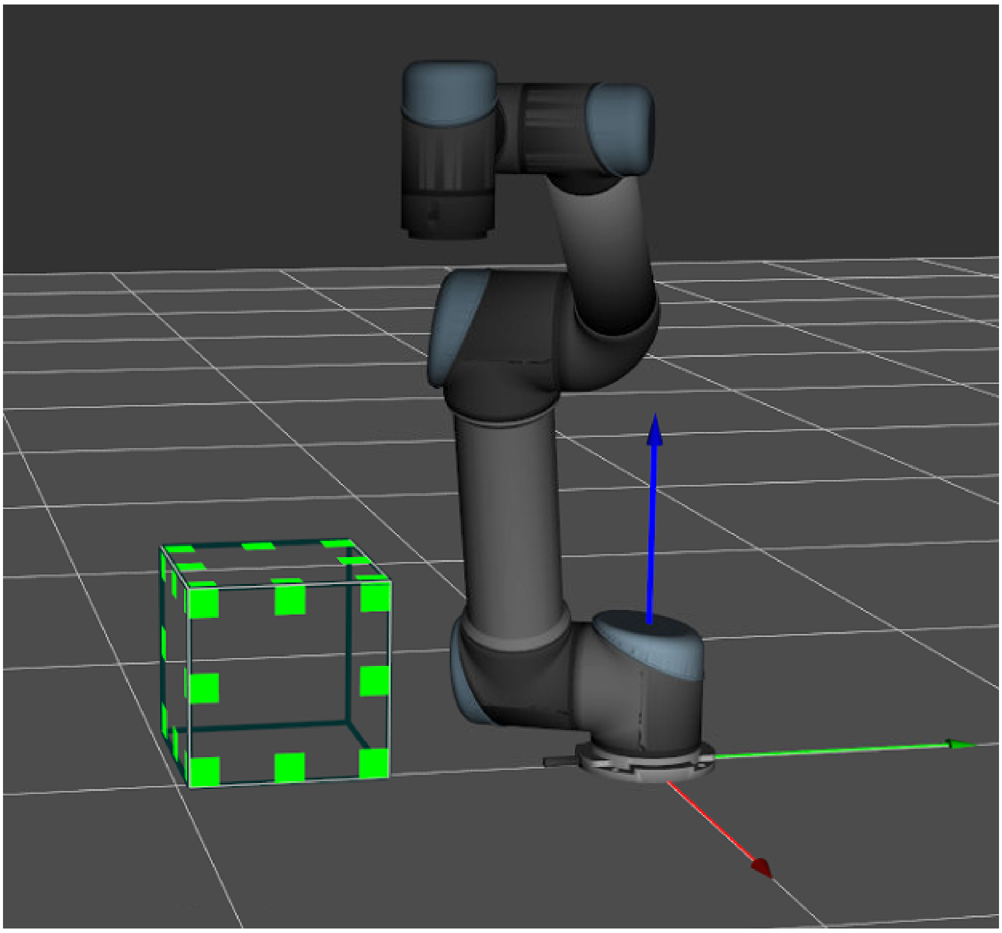
Z->Y'->X" ABB/KUKA/NACHI

Z 0.00°

Y' 0.00°

X" 0.00°

- Set Roi To Vision: Click this button, Mech-Viz will generate a “roiBoundary.json” file and save this file to corresponding vision project.
- Half Size: X, Y, Z are half the length of the side for the bounding box side in Region of Interest (ROI).
- Pose: The coordinates X, Y, Z, and Euler angles or quaternions are the center position of the Region of Interest (ROI).



Note: The Region of Interest (ROI) can be directly moved to the expected setting position by dragging the square part on the green border.

roiInFlange The edited ROI Boundary is defined in the flange coordinate system. If the selected vision project is eye in hand system, this parameter is checked.

4.8.5 visual_move

Description

Move to the pose which is dynamic pose got from Mech-Vision. It need to be used with *visual_look* .

Parameters

useAll For the trajectory scene, if it is True, all the poses given by vision service will be continuously executed at one time, instead of executing the *Pick-Hold-Place* of the next target when *Pick-Hold-Place* of one target is completed

If checked, it will show the parameter icon:

notMove If it is True, this module will actually participate in the planning (so avoid collisions with the path between the front and back positions), but this position will not be sent to the robot, and the results are generally only used for *relative_move* as a reference

If checked, it will show the parameter icon:

reuseVisionResult Reuse the remaining visual results from the previous execution.

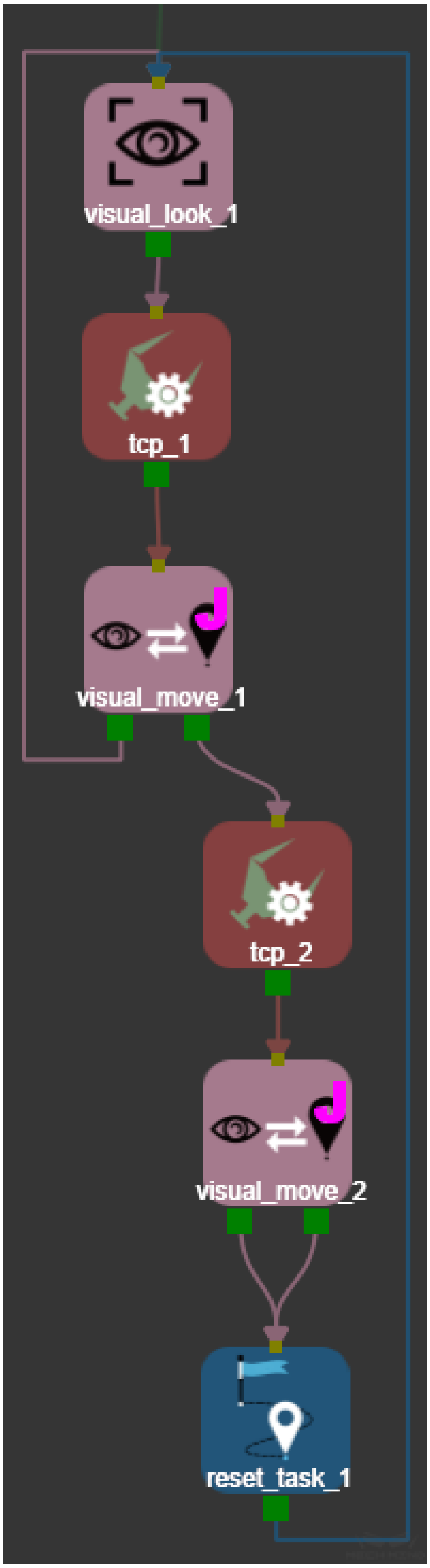
If checked, it will show the parameter icon:

reuseVisionResult If it is True, multiple *visual_move* will share the same vision result of one *visual_look* . Besides, you need to choose the Vision Name from *Get Vision Services* below.

Usage scenario: The object is not easy to be grasped. If picking failed first time, the robot will change anothor end effector to pick this object again by using the same vision result.

How to use:

1. As shown in the figure, one visual_look with multiple visual_move


visual_look_1
tcp_1
visual_move_1
tcp_2
visual_move_2
reset_task_1

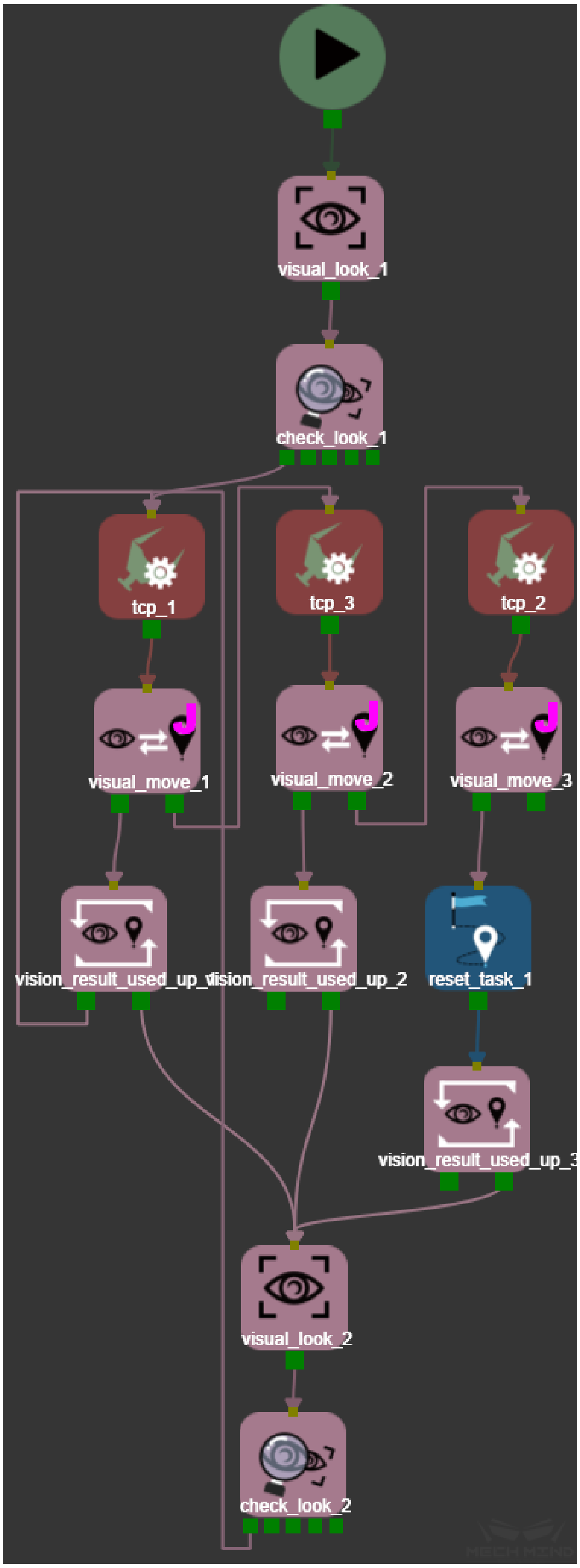
> **Attention:** It is important to connect a *reset_task* after the last *visual_move* to clear the shared vision result. Otherwise, it will use the previous vision result even if new detection is triggered, which will impact the running of project.

2. using with *reuseVisionResult*

    1. It must have picking pose before running the logic "switch end effectors then pick", otherwise it will pop up error when running. So you need to connect *check_look* after every *visual_look* , as shown below

    2. Connect *vision_result_used_up* after every visual_move and connect the reasonable processing both port 1 and 2 to ensure the project running as expect.

    3. Connect a *reset_task* after the last visual_move to finish the trying of "switch end effectors"


visual_look_1
check_look_1
tcp_1
tcp_3
tcp_2
visual_move_1
visual_move_2
visual_move_3
vision_result_used_up_1
vision_result_used_up_2
reset_task_1
vision_result_used_up_3
visual_look_2
check_look_2

Sucker Config It is applied to the scenario that the sucker size is bigger than box size, Mech-Viz can calculate the offset and decide whether it is a proper picking pose.

If the offset is calculated by Mech-Viz, it is necessary to use check collision function to avoid colliding with the nonstationary objects such as other picking objects. For the fixed objects in the scene, the collision checking should by added models, instead of point cloud. See the details: *Collision Detection* .

1. **suckerType**: Includes four types:

   1. Unspecified
   2. Continuous: continuous sucker, such as a sponge sucker.
   3. IndependentCtrl: combined sucker that can independently control the switching of multiple suckers.
   4. ContinuousWithMultiPick: continuous sucker which can pick multiple boxes per time. Based on type 2, it will pick boxes as much as possible according to the sucker size and boxes size, such as picking a row of boxes.

   If `Unspecified` is selected, Mech-Vision can provide the offset; if the remaining parameters is selected, Mech-Vision is not allowed to provide the offset. The offset is calculated by Mech-Viz and combined with *Collision Detection* to determine whether it can be grasped.

2. **offsetStrategy**: *sucker bias principle*
3. **suckerLenX**: x-direction length of sucker.
4. **suckerLenY**: y-direction length of sucker.
5. **boxNotExceedSucker**: If it is True, the combined boxes will not be picked once its size bigger than sucker size
6. **suckerConfigPath**: Path of the sucker configuration file in IndependentCtrl mode, see *Individual control mode for combined suction cup*

Avoid Pick Same Pose

1. **samePoseDistThre**: If the spatial straight distance between a pose and the last grasping pose is less than this threshold, they are considered to be the same pose. If it is 0, it indicates that the function of *Avoid Pick Same Pose* is not used.
2. **poseDiscardDistThre**: If the spatial straight distance between this pose and the last grasping pose is smaller than this threshold, then give up the grasping.

   This parameter is always used with *samePoseDistThre* to solve the problem that failed picking leads process stopping due to overweighed objects. If the first time picking failed and the position of this object changed, the priority of this pose will be reduce, robot will pick this objects again after picking other objects; but if the position has not changed when failed picking, this pose will be delete.

   Therefore, the value of *poseDiscardDistThre* should smaller than *samePoseDistThre*

3. **sizeOfRecorded**: Record the maximum number of poses or objects.
4. **recordType**:

   - PickPose: Record by poses. For example, the object X has three picking pose, if one of them has been tried, the other two poses still can be picked.

- Object: Record by object. For example: if one of the object X' s poses has been tried, the other two poses can not be picked anymore, robot will try to pick other objects next time.

MultiPick

1. **isMultiPick**: If it is true, it will use the traditional multi-picking mode.
2. **multiPickConfigPath**: File path for multi-grasping configuration.
3. **distThre**: The distance deviation threshold between two adjacent boxes when combine several small boxes to a picking unit. The boxes cannot be combined when the distance larger than this value.
4. **angleThre**: The angle deviation threshold around Z axis of each box when combine several small boxes to a picking unit. The boxes cannot be combined when the angle larger than this value.

PickingCount

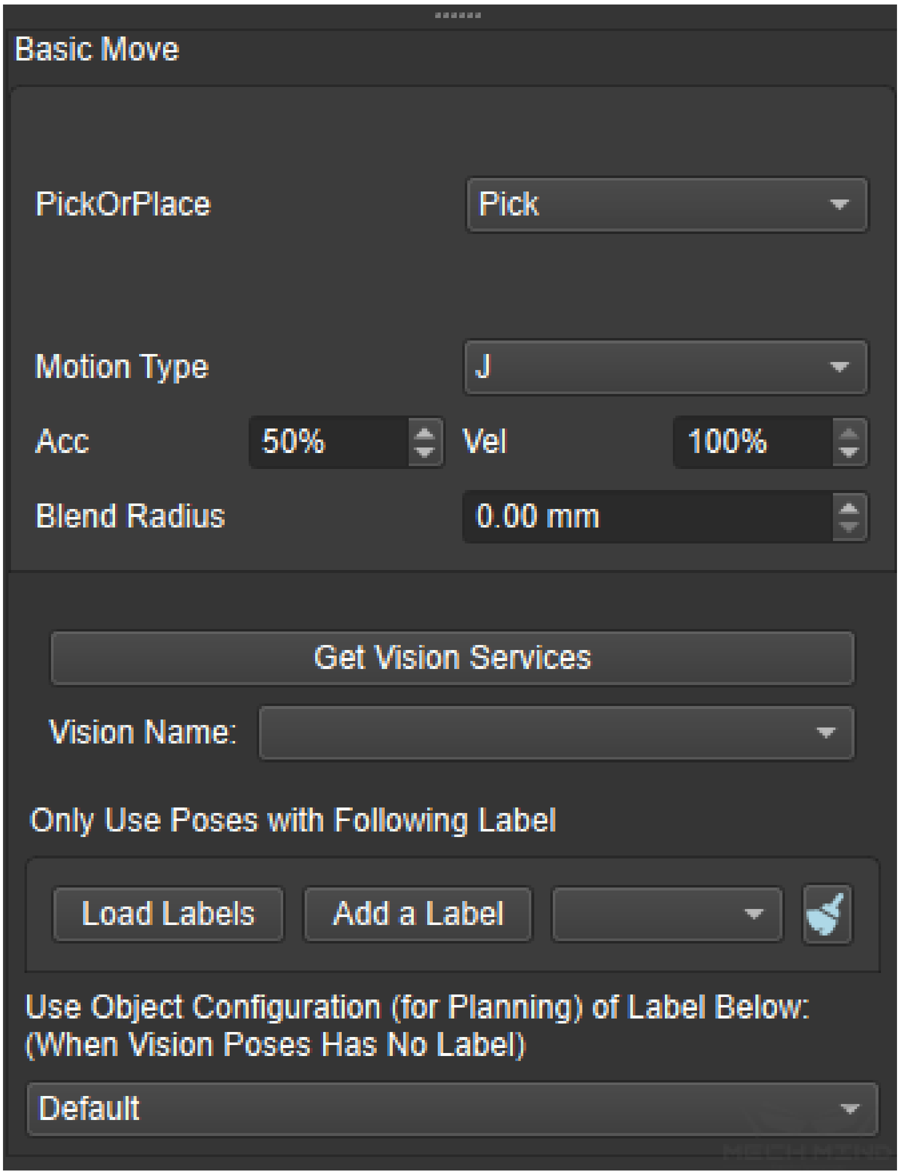
1. **expectedPickingCount**: Preset the total number of target grasped. Stop when the number of target grasped is reached.
2. **totalPickedObjCount**: Cumulate the total number of grasped. It is no editable
3. **pickedObjCount**: Number of current grasped; the actual number of grasped for this time, it is no editable.

Symmetry xyRotLeastChange: If it is checked, robot will move to the pose which the rotation around x/y axis in JPs or TCP is minimal, comparing with the previous *move* .

saveVisionResult If it is checked as True, the actual executed pose of each planning result will be recorded in the succeed_vision_records next to the execution file in the root directory of Mech-Viz.

pickBoxName The picking box model name when update the model by vision service.

Basic Move



Basic Move

PickOrPlace: Pick

Motion Type: J

Acc: 50%

Vel: 100%

Blend Radius: 0.00 mm

Vision Name:

Only Use Poses with Following Label

Use Object Configuration (for Planning) of Label Below:
(When Vision Poses Has No Label)

Default

See *Basic Move General Parameters*

Get Vision Services

Select the vision project name which is needed to get poses. These function can be achieved:

1. Multiple visual_move can share the same vision result

#. Assign some visual_move getting vision result from certain vision service(visual_look). It is not necessary to using the combination like [one *visual_look* must connect one *visual_move*]

Only Use Poses with Following Label

Load Labels When select a certain label, this visual_move will only pick the object with this label.

Use Symmetry of

It will acquire all the labels which are edited in *Configuration of Objects to Pick (for Move Planning)*. If Mech-Vision gives poses with labels, this module will use the corresponding symmetry with the given label; if there is no label given by Mech-Vision, this module will use the symmetry according to the selected label in this parameter.

sucker bias principle

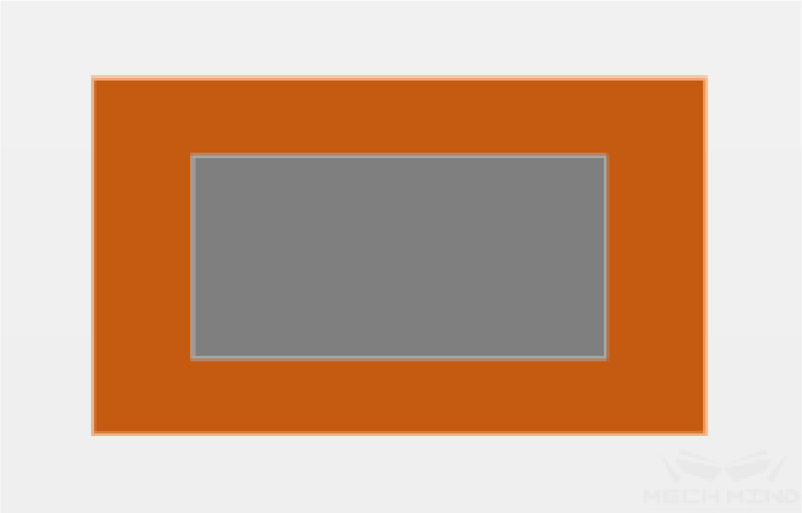
Select *CenterPrior* , try the pose without offset first (the center of the sucker is aligned with the center of the box), try an offset (center misalignment) pose after the path planning fails; CornerPrior is the opposite, and it will try the unbiased pose at last. The relationship between the relative size of the sucker and the box is divided into the following five cases, and there are 1 to 3 types of grasping postures in each case. In different cases, the preferred sequence of attempts for each posture type is as follows.

Attention: CornerPrior is applied to mixed stacking to reduce collision angles, so if the placement after grabbing is not mixed stacking (such as on a conveyor belt), CornerPrior is not required.

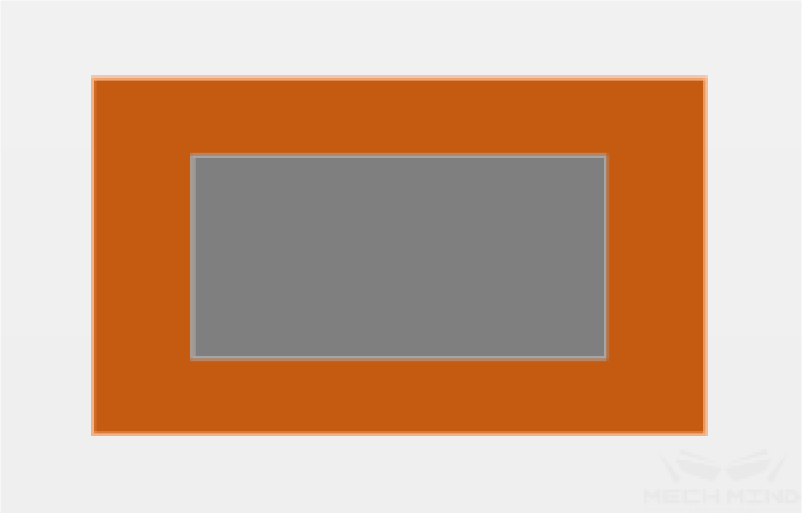
Note: In the following illustration, orange is the box, and gray is the sucker.

Scene 1:

1. sucker short side < box short side < sucker long side < box long side
2. sucker short side < sucker long side < box short side < box long side



Actual grasping position:



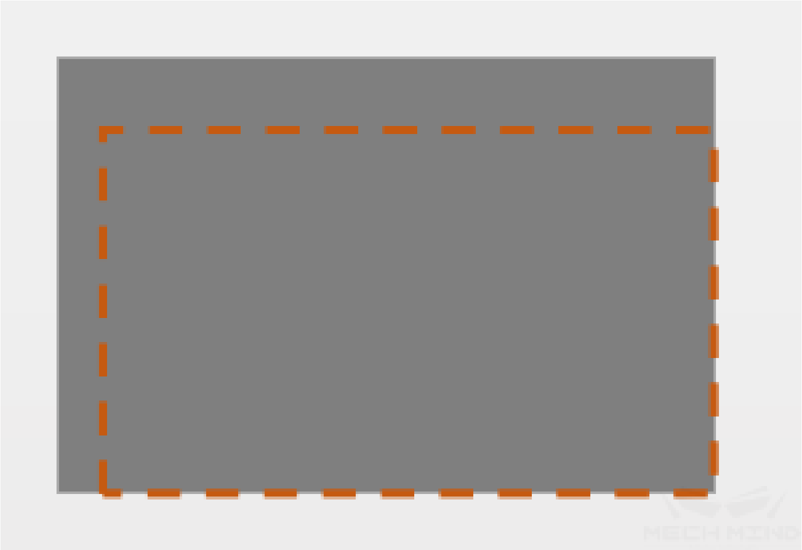
Scene 2:

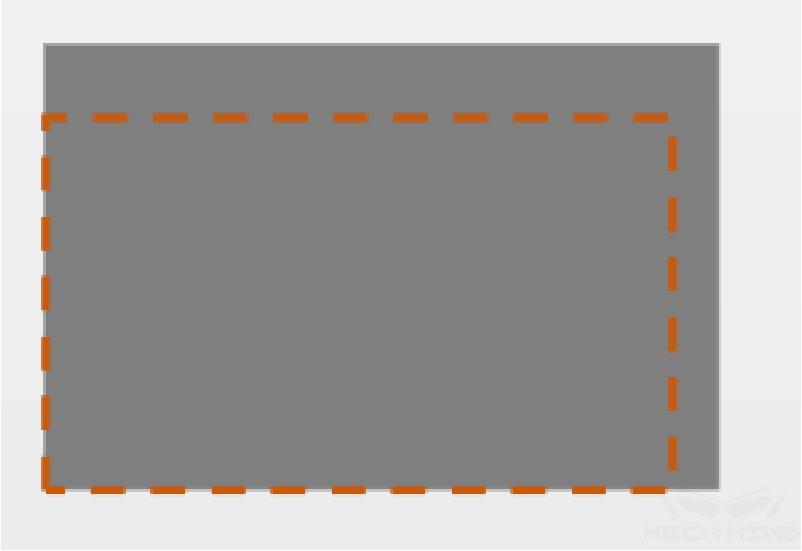
sucker short side < box short side < sucker long side < box long side

Actual grasping position:

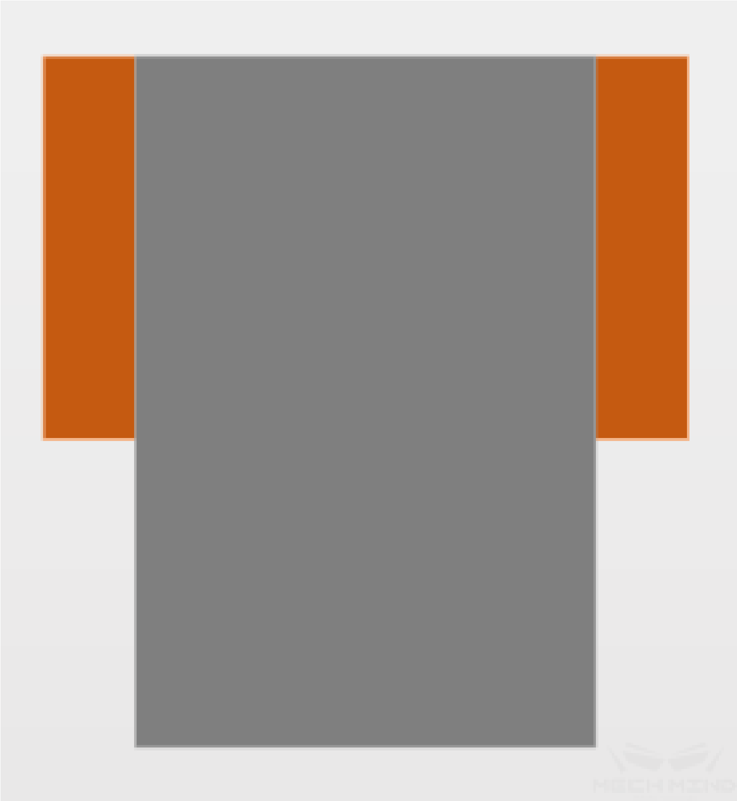
Method 1:

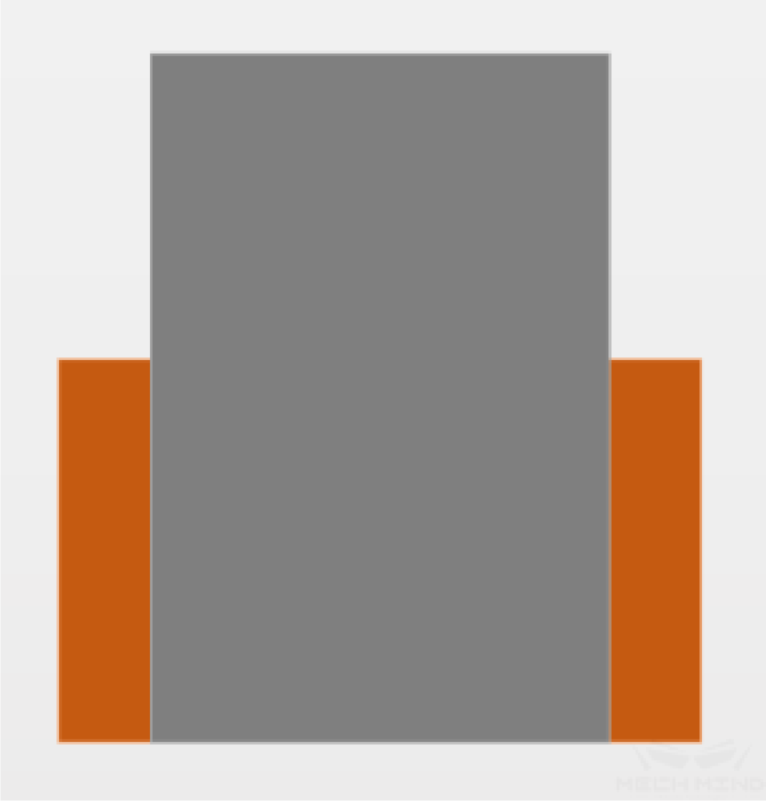
Method 2:





Method 3:





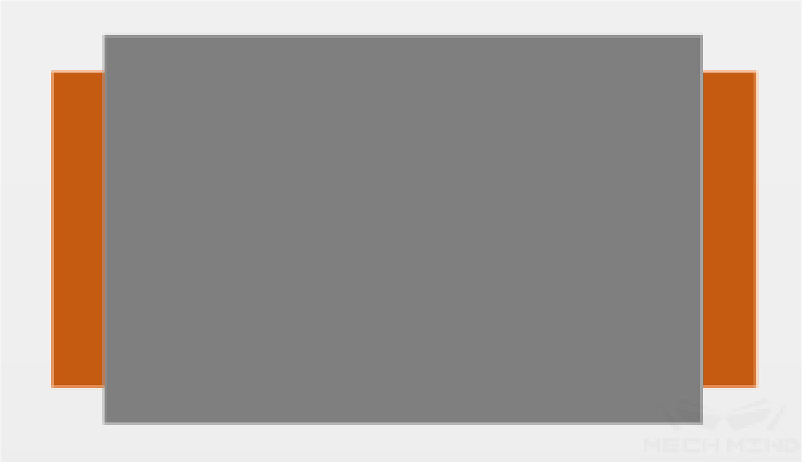
1. When CenterPrior is selected, the grasping method order is: 1 -> 2 -> 3 (If the ratio of the sucker

short side to the box long side is less than 0.25, then: 1-> 2)

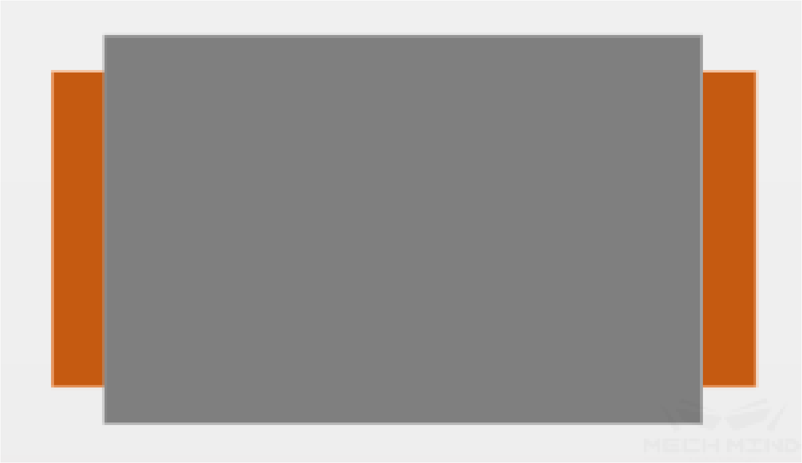
2. When CornerPrior is selected, the grasping method order is: 3 -> 2 -> 1 (If the ratio of the sucker short side to the box long side is less than 0.25, then: 2-> 1)

Scene 3:

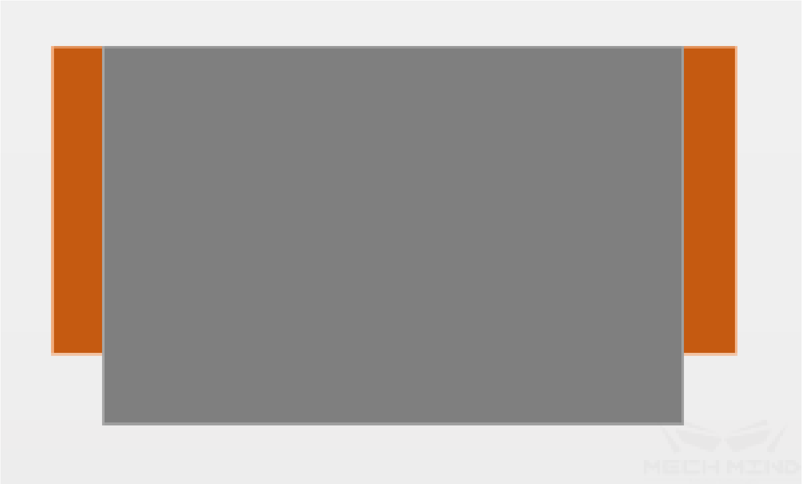
Box short side < sucker short side < sucker long side < box long side

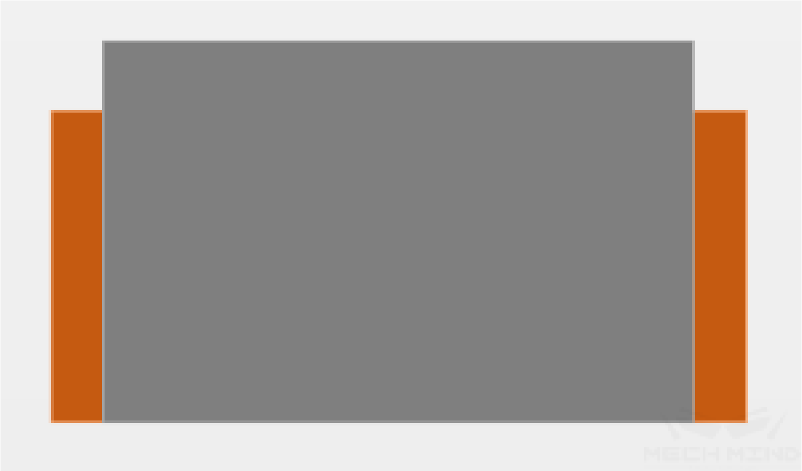


Method 1:



Method 2:

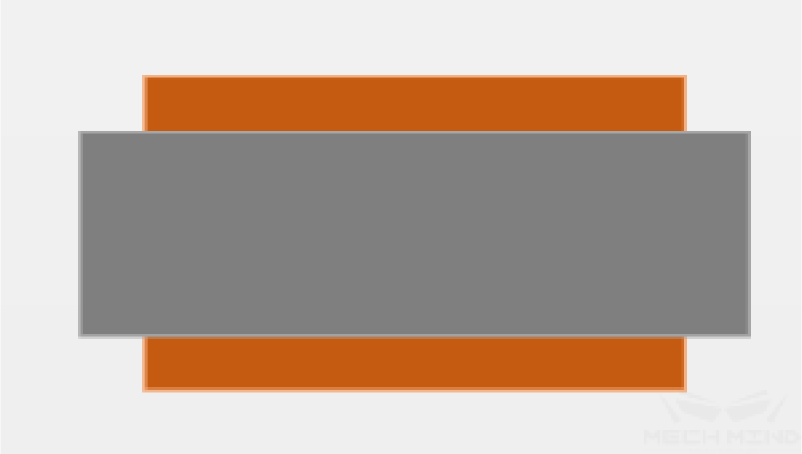




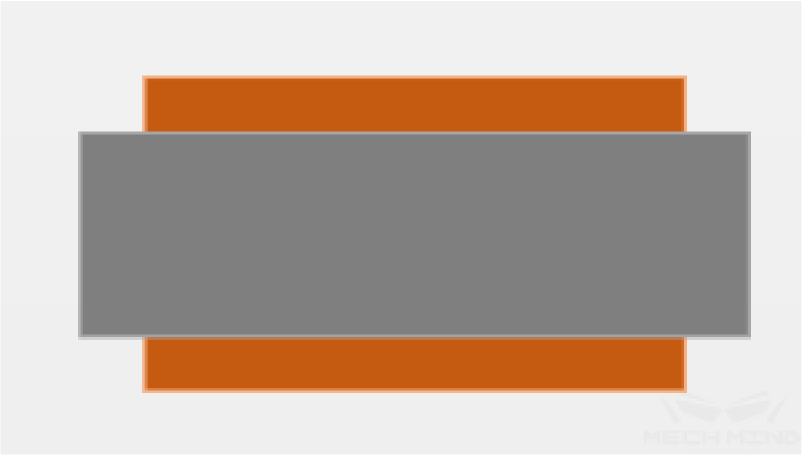
1. When *CenterPrior* is selected, the grasping method order is: 1 -> 2
2. When *CornerPrior* is selected, the grasping method order is: 2 -> 1

Scene 4:

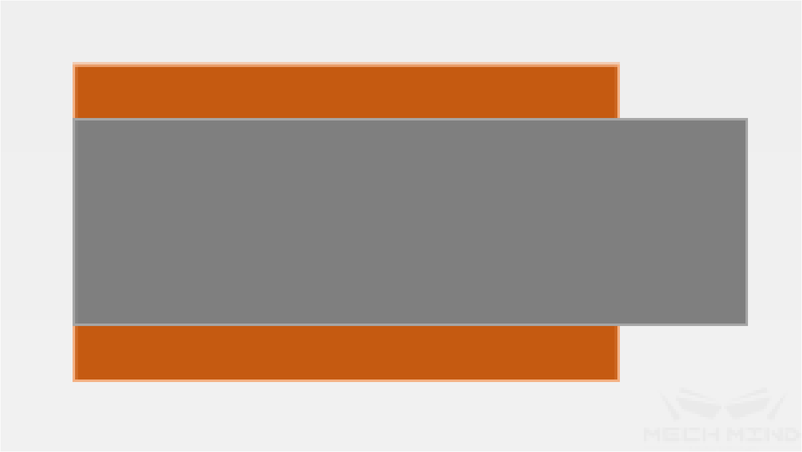
sucker short side < box short side < box long side < sucker long side

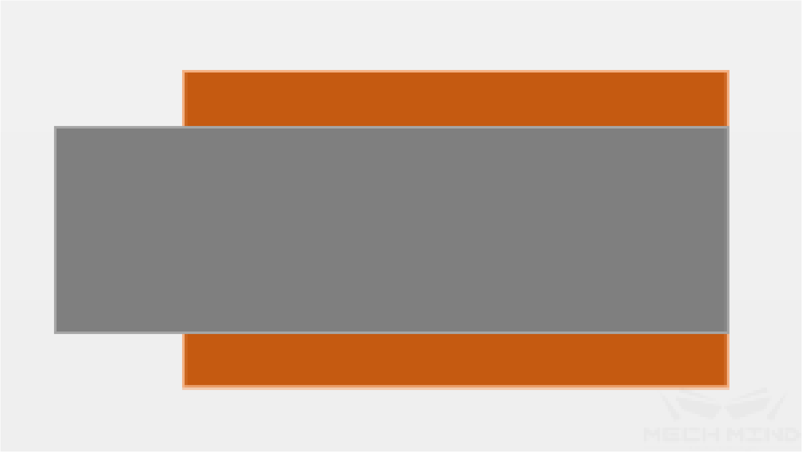


Method 1:



Method 2:

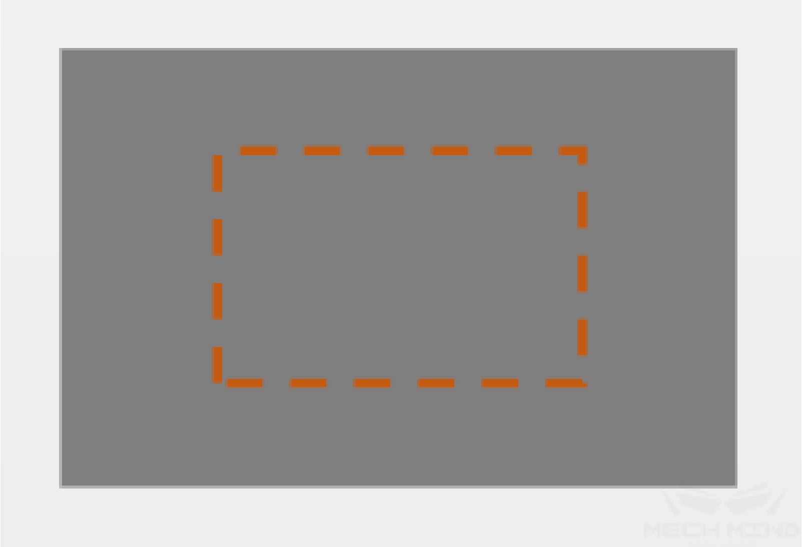




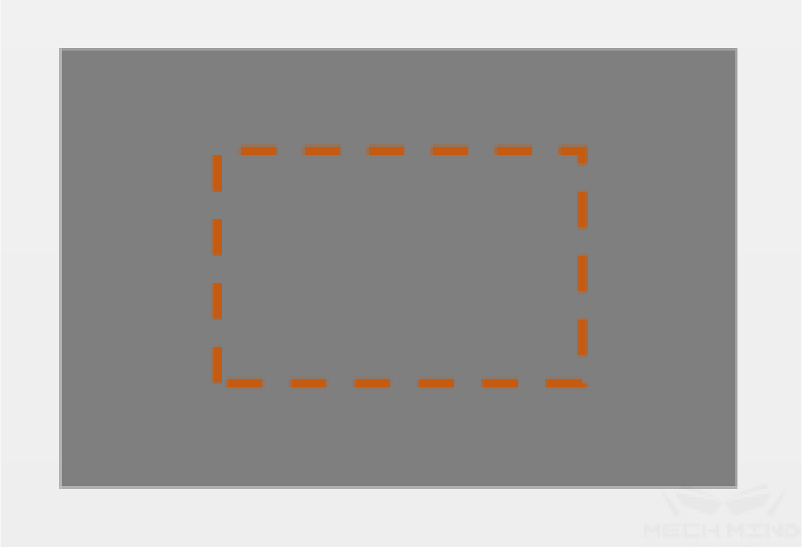
1. When CenterPrior is selected, the grasping method order is: 1 -> 2
2. When CornerPrior is selected, the grasping method order is: 2 -> 1

Scene 5:

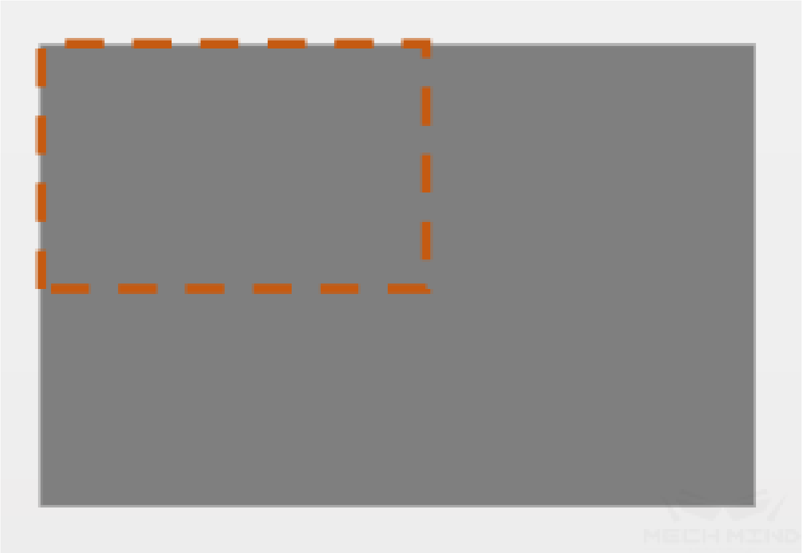
Box short side < box long side < sucker short side < sucker long side

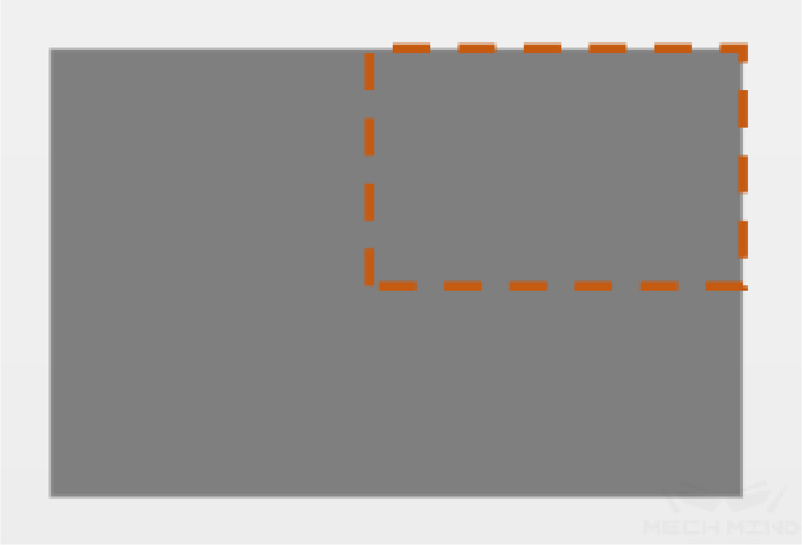


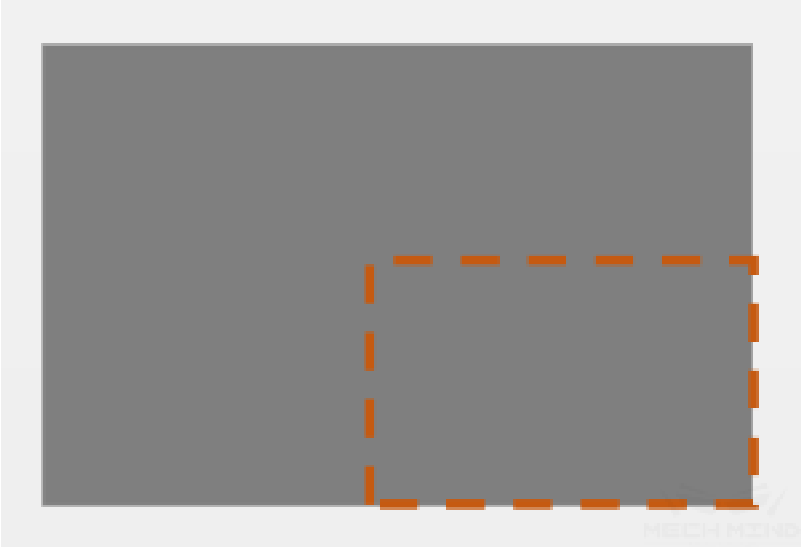
Method 1:

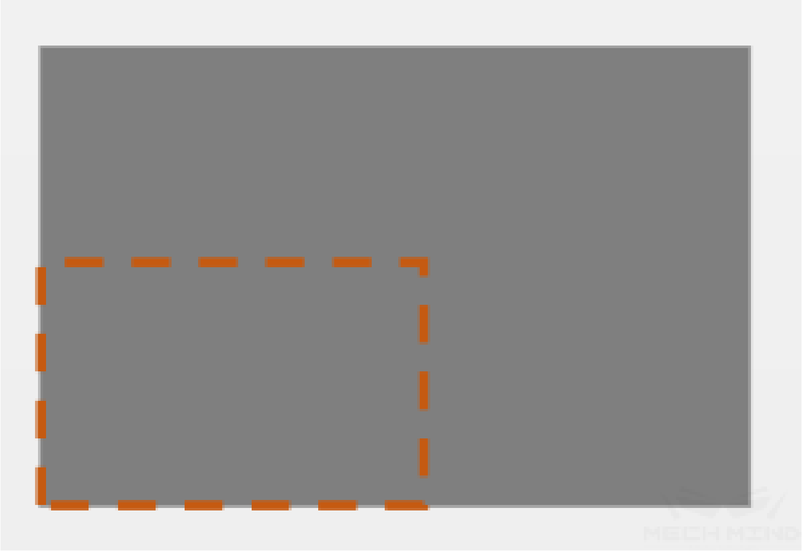


Method 2:









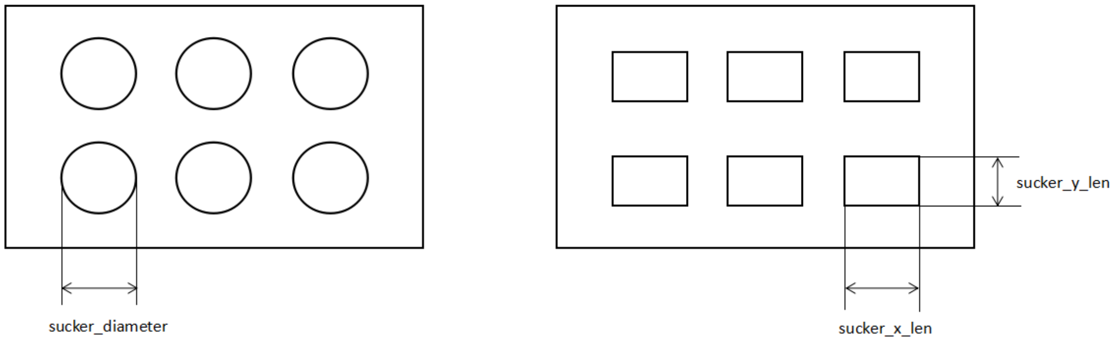
1. When CenterPrior is selected, the grasping method order is: 1 -> 2
2. When CornerPrior is selected, the grasping method order is: 2 -> 1

Individual control mode for combined suction cup

The individual control mode for combined suction cup supports a single suction cup unit as a circle or a rectangle. The parameters of suction cup are written in the suction cup configuration file. The configuration file is json file format, which includes information about the size of a single suction cup, the number of rows and columns of the combined suction cup, the corresponding DO port and etc.

If the suction cup is round, use sucker_diameter as the suction cup diameter;

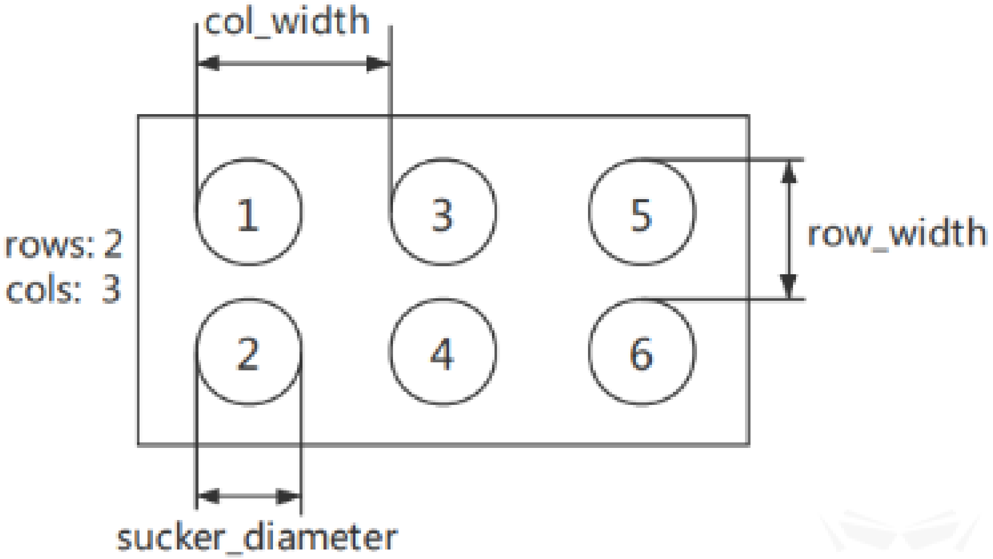
If the suction cup is rectangular, use sucker_x_len and sucker_y_len to indicate the length in the x and y directions of a single suction cup unit in the tcp coordinate system respectively.

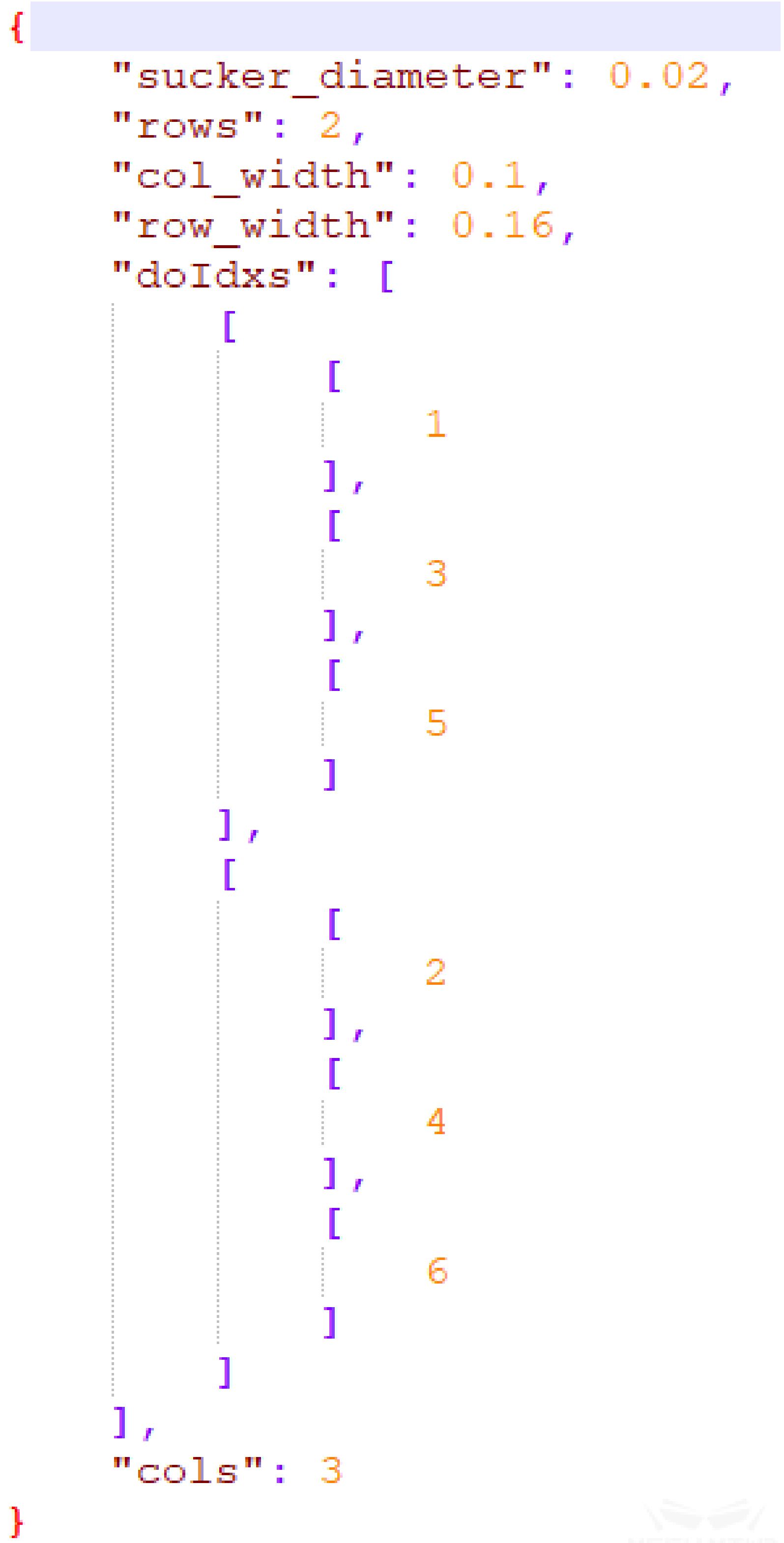


圆形吸盘 矩形吸盘

Attention: When the suction cup offset is calculated by Mech-Viz, the picking pose will be recalculated. In IndependentCtrl mode, the object must be strictly larger than the smallest suction cup unit, otherwise the software will not calculate the visual grasping point and report a “no pose” error.

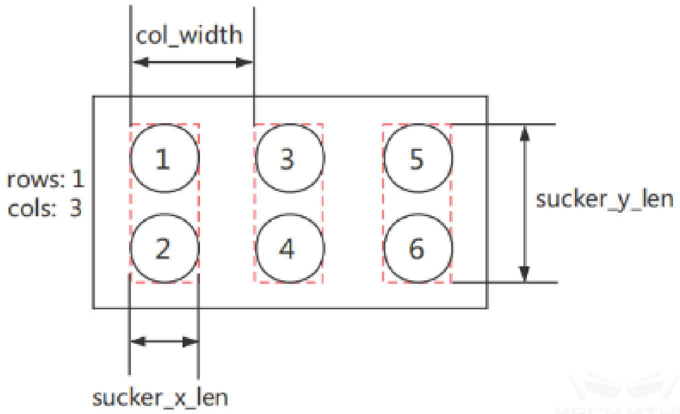
The combination of 6 round suction cups as an example. When a single suction cup can be independently controlled, the configuration file is as shown in the figure. The number in doIdxs is the DO port number of the actual control

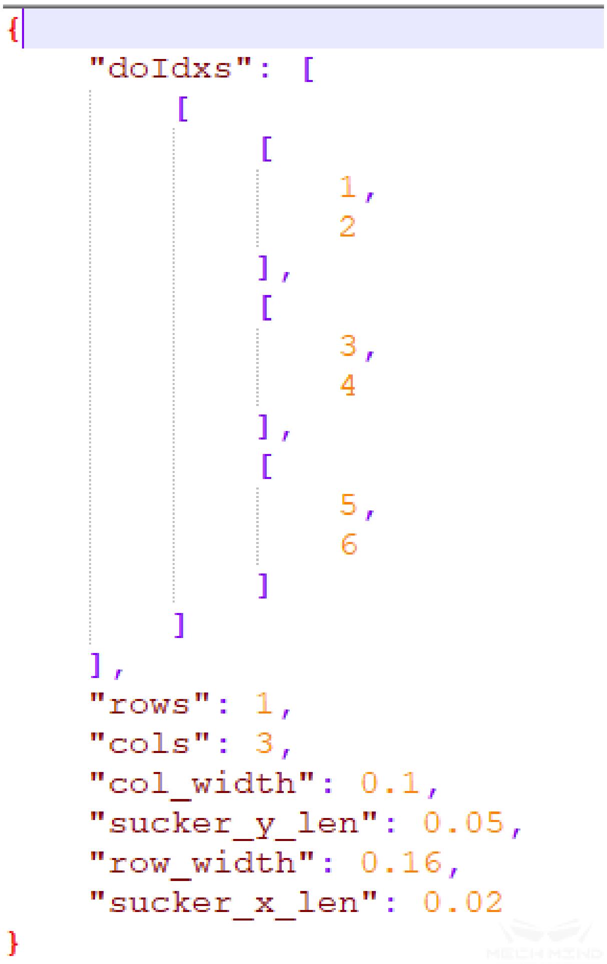




```
{
    "sucker_diameter": 0.02,
    "rows": 2,
    "col_width": 0.1,
    "row_width": 0.16,
    "doIdxs": [
        [
            [
                1
            ],
            [
                3
            ],
            [
                5
            ]
        ],
        [
            [
                2
            ],
            [
                4
            ],
            [
                6
            ]
        ]
    ],
    "cols": 3
}
```

In practical applications, the suction cups cannot be controlled independently, and some small combinations can be made. The following figure is an example. It is not practical to use only a single suction cup to grasping the box. You can only use the combination of suction cups by pressing the red frame. The suction cup can be regarded as the smallest unit at this time, and it changes from two rows and three columns to one row and three columns. The corresponding configuration file is shown in the figure.

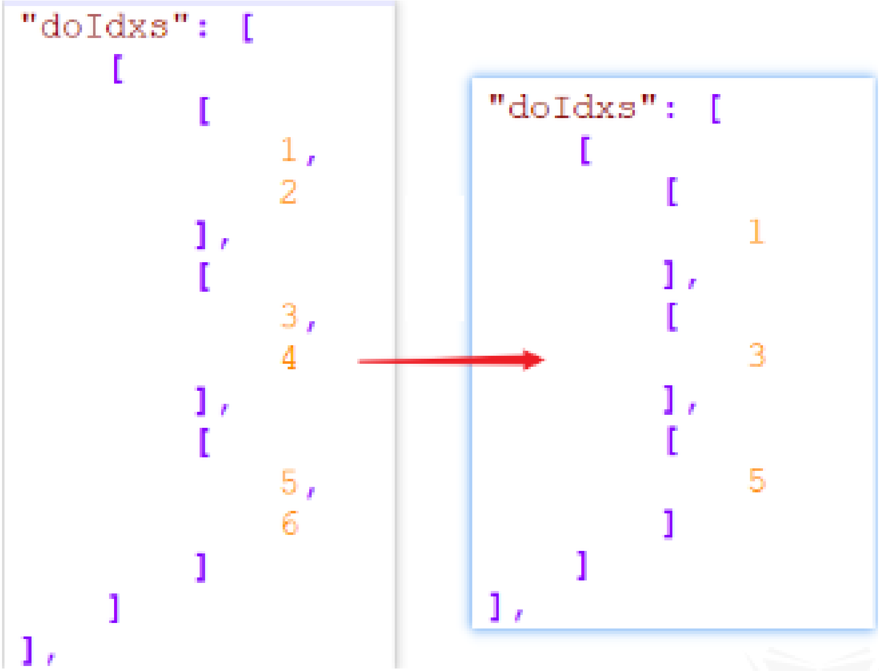




```
{
    "doIdxs": [
        [
            [
                1,
                2
            ],
            [
                3,
                4
            ],
            [
                5,
                6
            ]
        ]
    ],
    "rows": 1,
    "cols": 3,
    "col_width": 0.1,
    "sucker_y_len": 0.05,
    "row_width": 0.16,
    "sucker_x_len": 0.02
}
```

Note: Currently, only the adjacent suction cups are allowed to combined into rectangular suction cup units. Other special combinations are not supported.

If DO port 1 can control suction cups 1 and 2, DO port 3 can control suction cups 3 and 4, and DO port 5 can control suction cups 5 and 6, the editable json file is as follows:



```
"doIdxs": [
    [
        [
            1,
            2
        ],
        [
            3,
            4
        ],
        [
            5,
            6
        ]
    ]
],
```

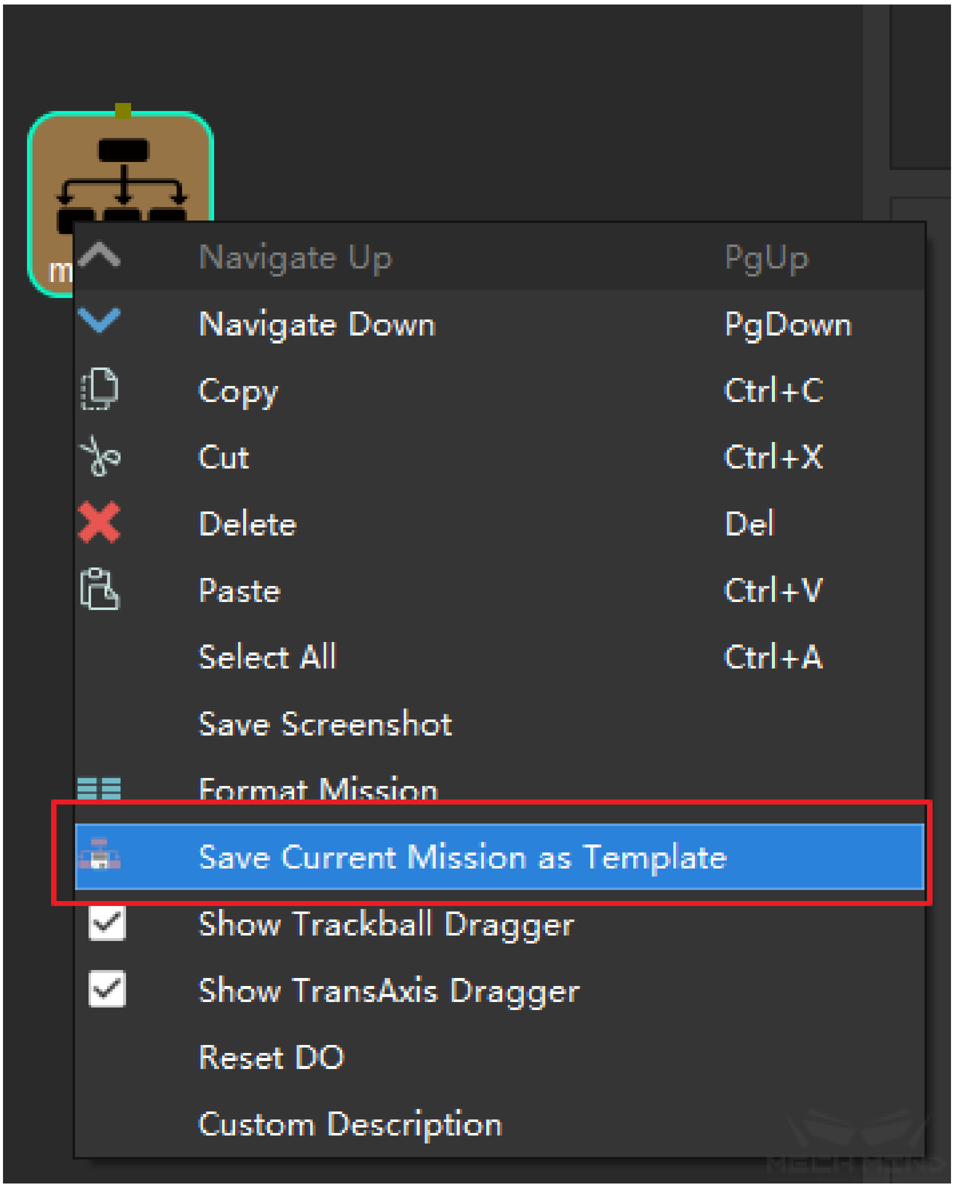
```
"doIdxs": [
    [
        [
            1
        ],
        [
            3
        ],
        [
            5
        ]
    ]
],
```

4.9 Custom

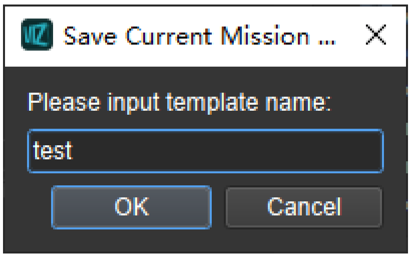
This type saves the programs that users need in the form of mission; for example, when a part of the program is more commonly used or is reused in this project, you may save this part of the program and drag it out from the user-defined class directly when it is used in next time.

Examples of usage:

As shown in the programming area, *right-click* → *Save Current Mission*

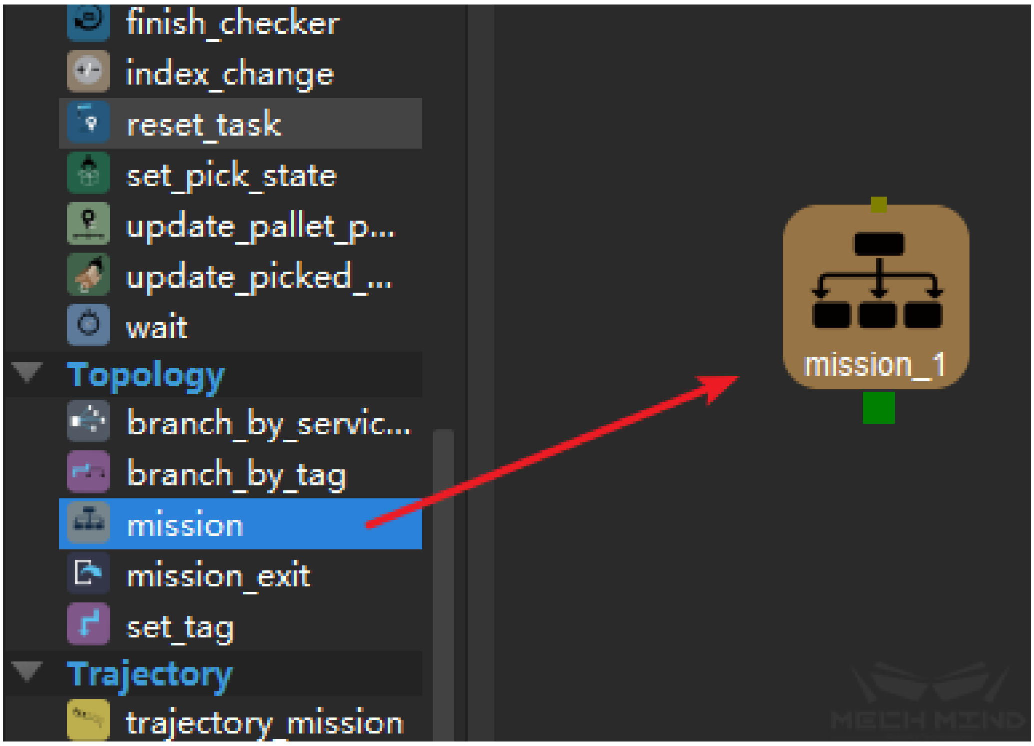


Name the module according to the function of the current program


Save Current Mission ...

Please input template name: test

After saving, this program will appear as a mission module in the user-defined category in the module area. When it needs to be used again, drag it to the programming area.



4.10 Others

4.10.1 get_jps

Description

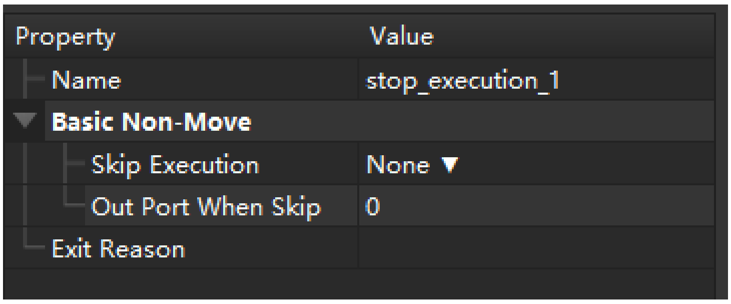
When Mech-Viz does not fully control all robot movements, it is required to obtain the current joint angle and continue to control the robot movement after the robot moved autonomously.

4.10.2 stop_execution

Description

For the complex usage scenerio, like multi-layer-nested structure: a *Multi-Level Mission Processing* nests a *Multi-Level Mission Processing* , etc. If there has no task to be executed in a inner layer, and the whole project need to be stopped, you can connect this module

Parameters



Property	Value
Name	stop_execution_1
Basic Non-Move	
Skip Execution	None ▼
Out Port When Skip	0
Exit Reason	

exitReason You can describe the reason of stop briefly

4.11 Pallet

Palletizing related modules are divided into five according to different application requirements. Users can choose one of them to connect according to the requirements.

smart_pallet_pattern - Choose common pallet patterns for palletizing

general_visual_pallet - Finish palletizing based on visual recognition results

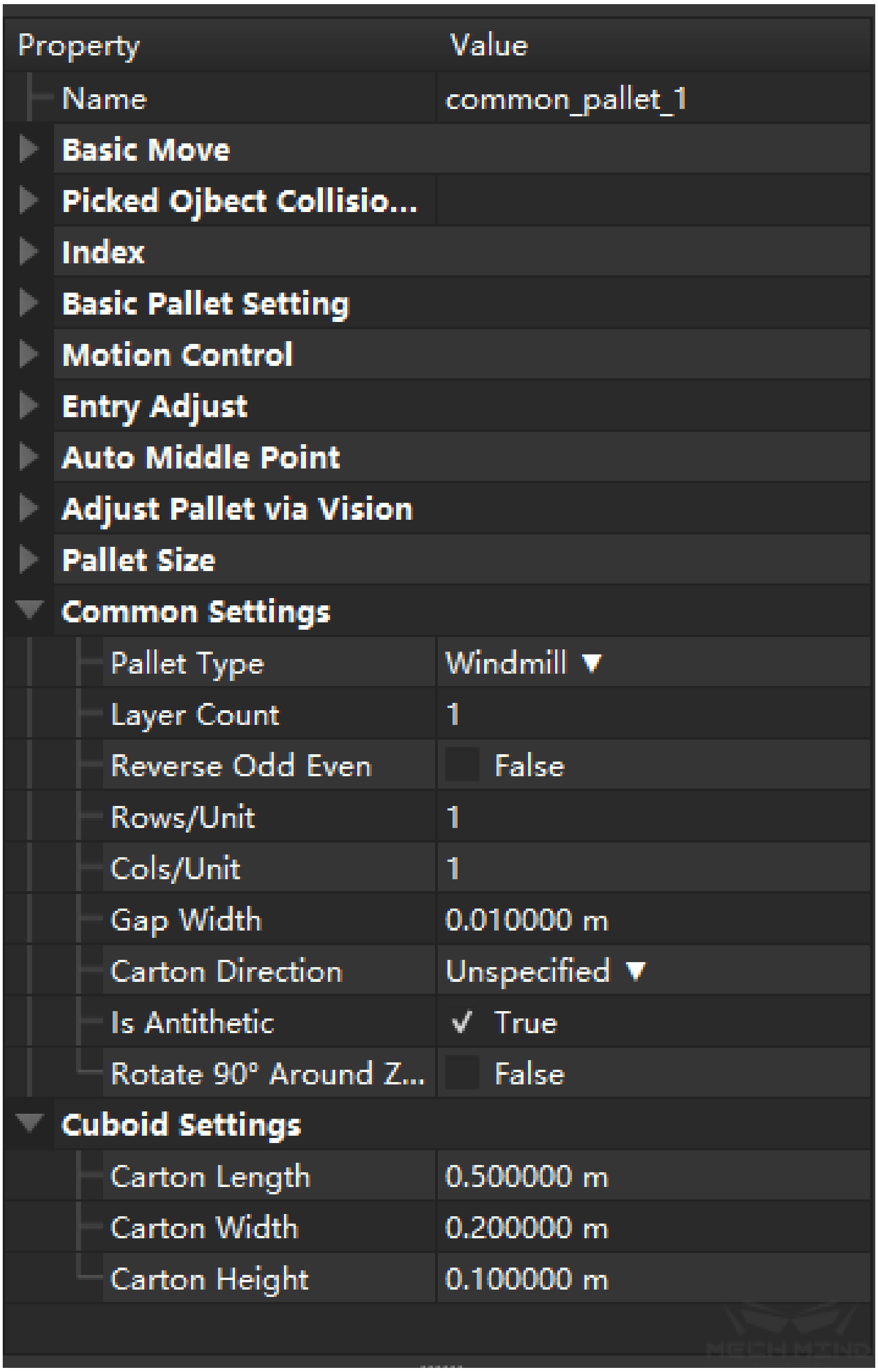
mixed pallet - Palletizing objects with different sizes

custom_pallet_pattern - User-defined stacking rules

optimal_pallet - Plan the optimal pallet pattern according to the pallet and box size

4.11.1 smart_pallet_pattern

Description



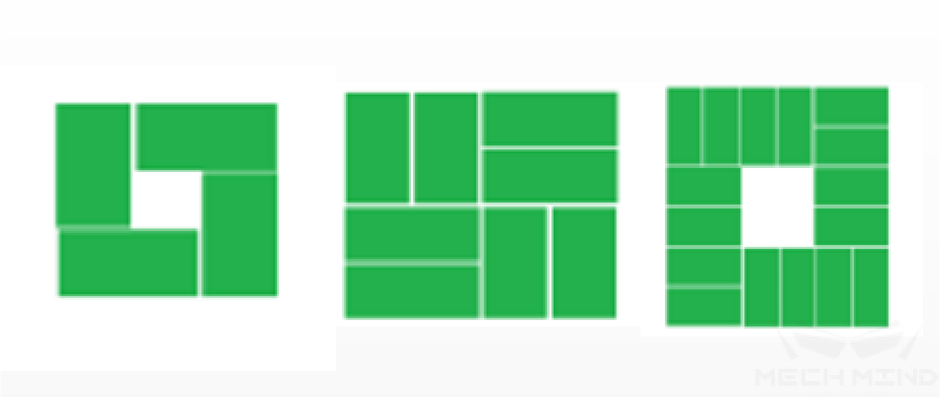
Property	Value
Name	common_pallet_1
▶ Basic Move	
▶ Picked Ojbect Collisio...	
▶ Index	
▶ Basic Pallet Setting	
▶ Motion Control	
▶ Entry Adjust	
▶ Auto Middle Point	
▶ Adjust Pallet via Vision	
▶ Pallet Size	
▼ Common Settings	
Pallet Type	Windmill ▼
Layer Count	1
Reverse Odd Even	□ False
Rows/Unit	1
Cols/Unit	1
Gap Width	0.010000 m
Carton Direction	Unspecified ▼
Is Antithetic	✓ True
Rotate 90° Around Z...	□ False
▼ Cuboid Settings	
Carton Length	0.500000 m
Carton Width	0.200000 m
Carton Height	0.100000 m

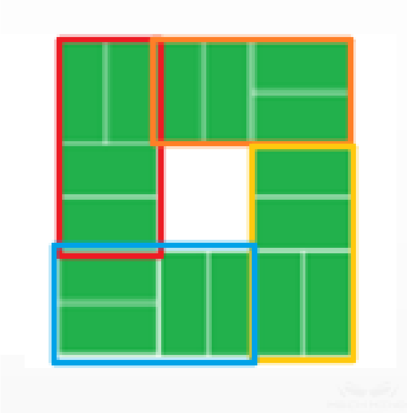
In addition to *Parameters* , smart_pallet_pattern module has settings for palletizing type and object size

Parameters

Common Settings

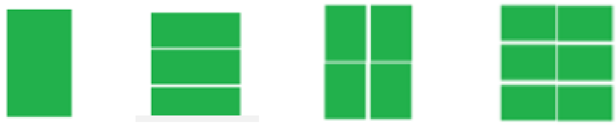
- reverseOddEven: The default is False. When True is selected, the swap the parity layer
- palletPattern:
  - Windmill: as shown in the figure





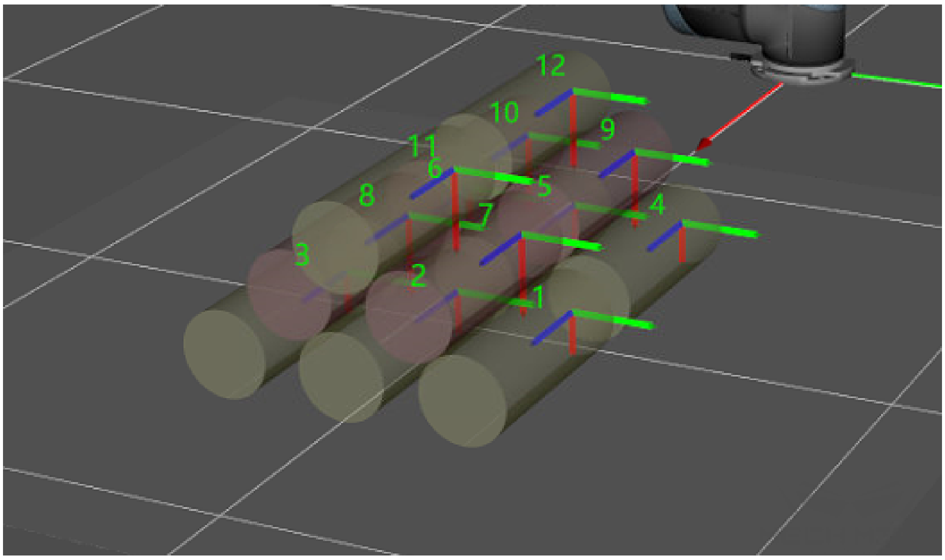
Different color boxes in the figure represent a unit. The user can configure total rows and columns in each unit

  - Grid: as shown in the figure



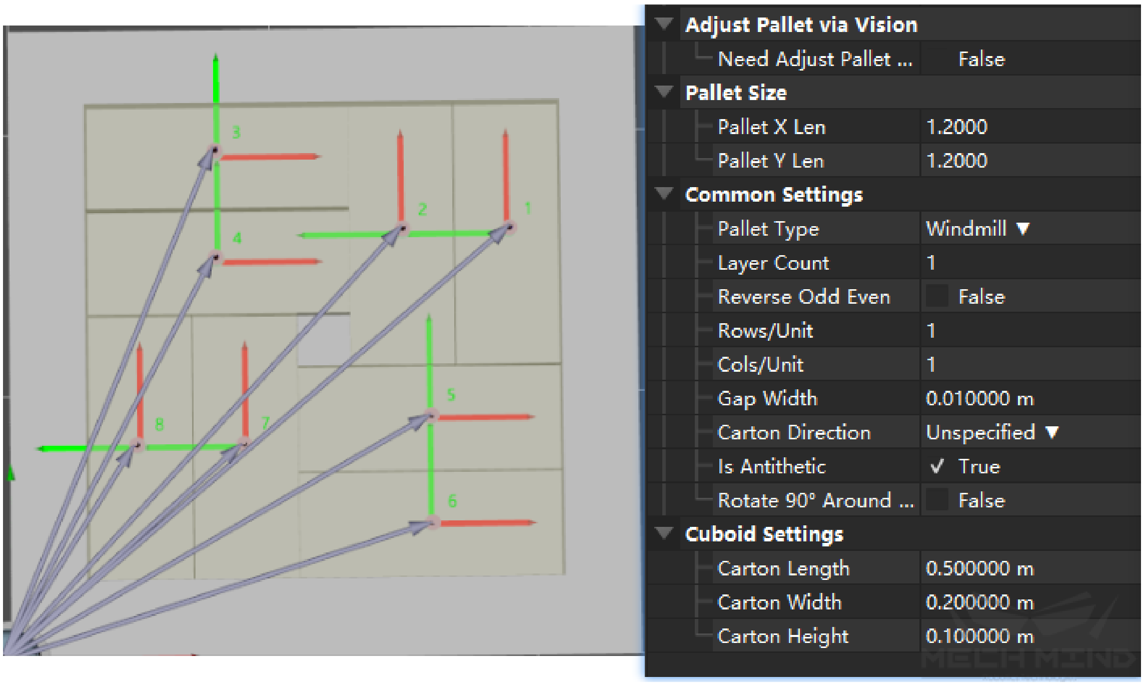
The user can configure total rows and columns in each unit

  - Cylinder: it is used for cylindrical palletizing and can be configured in *CylinderInfos*



- rowsPerUnit: Set the row number of the box in a unit
- colsPerUnit: Set the column number of the box in a unit
- layerNum: Set the number of layer for palletizing
- palletLength: Set the length of the pallet, the unit is m
- palletWidth: Set the width of the pallet, the unit is m
- gapWidth: Set the spacing between each box, the unit is m
- cartonDirection: It is only applicable to windmill pallet pattern. If the object is a sack with an opening, it may specify the opening orientation; you may choose Unspecified, Inward or Outward.
- isAntithetic: It only takes effect when the pallet pattern is square; if True is selected, the parity layers use different placement layouts to enhance the stability of the stack; if False is selected, parity layers are placed in the same layout. If the pallet pattern is not square, even if it is hooked, it is impossible to use different placement of the parity layer.
- adjustToSquare: The default is False; if True is selected, even if the pallet pattern is not square, you may adjust the pallet pattern to square by adjusting the gap of the boxes. It is usually used in conjunction with *isAntithetic*.

CartonInfos This set of parameters is only used for visual display when setting the stacking parameters, as shown in the figure.



- **Adjust Pallet via Vision**
  - Need Adjust Pallet ...: False
- **Pallet Size**
  - Pallet X Len: 1.2000
  - Pallet Y Len: 1.2000
- **Common Settings**
  - Pallet Type: Windmill ▼
  - Layer Count: 1
  - Reverse Odd Even: □ False
  - Rows/Unit: 1
  - Cols/Unit: 1
  - Gap Width: 0.010000 m
  - Carton Direction: Unspecified ▼
  - Is Antithetic: ✓ True
  - Rotate 90° Around ...: □ False
- **Cuboid Settings**
  - Carton Length: 0.500000 m
  - Carton Width: 0.200000 m
  - Carton Height: 0.100000 m

When running the program, the relevant information of the box is inputted by external service, and the manual setting is invalid at this time.

CylinderInfos This set of parameters is used to set the shape parameters of the cylindrical object, the number of stacking rows and columns, and the spacing between the rows/columns, etc.

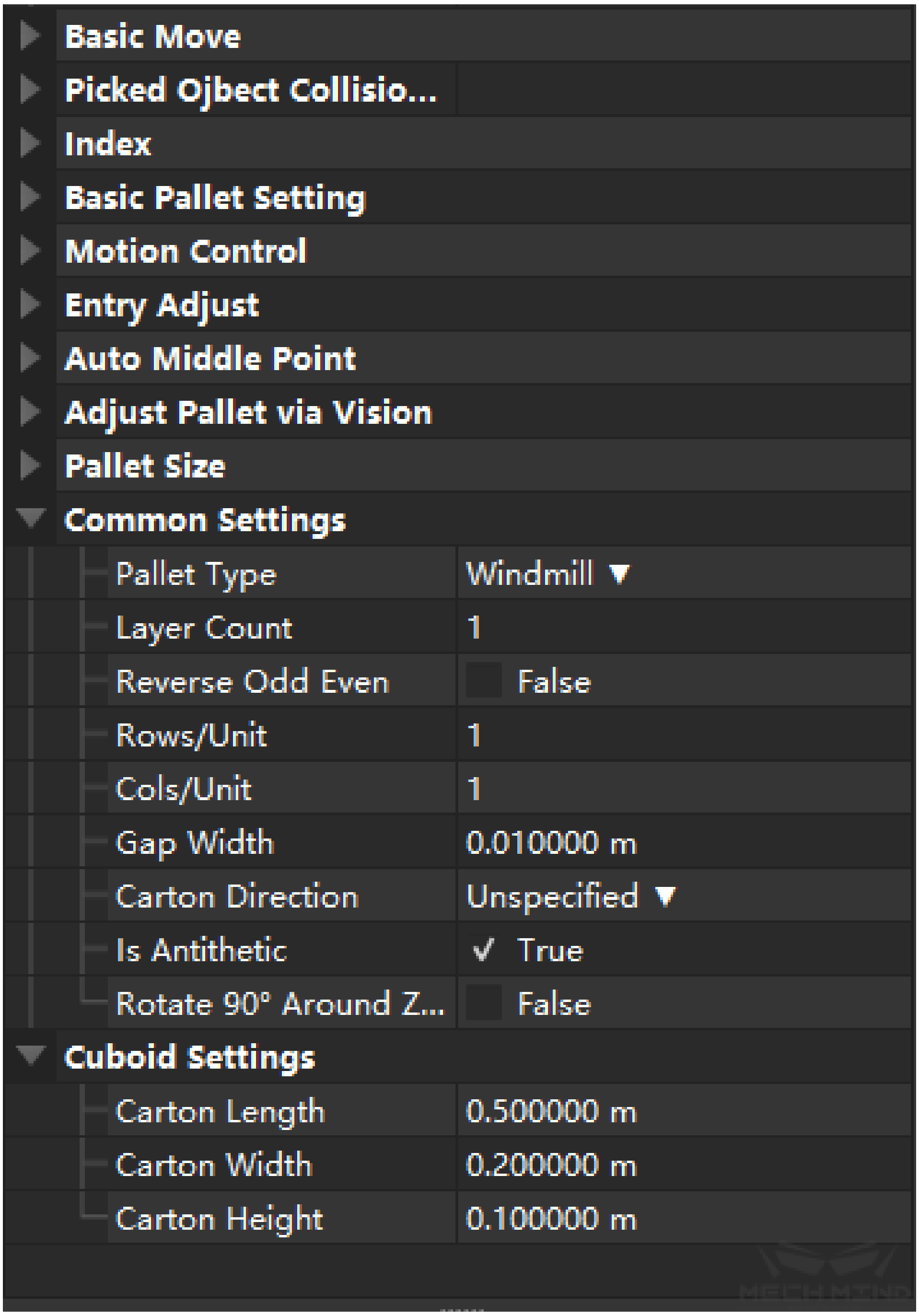
4.11.2 general_visual_pallet

4.11.3 mixed pallet

Description

mixed pallet is used to stack objects with different sizes. The pallet pattern can be automatically generated according to the parameters set by user and the size of the target box.

Parameters



Basic Move

Picked Ojbect Collisio...

Index

Basic Pallet Setting

Motion Control

Entry Adjust

Auto Middle Point

Adjust Pallet via Vision

Pallet Size

Common Settings

Pallet Type	Windmill ▼
Layer Count	1
Reverse Odd Even	□ False
Rows/Unit	1
Cols/Unit	1
Gap Width	0.010000 m
Carton Direction	Unspecified ▼
Is Antithetic	✓ True
Rotate 90° Around Z...	□ False

Cuboid Settings

Carton Length	0.500000 m
Carton Width	0.200000 m
Carton Height	0.100000 m

In addition to *Parameters* , the following parameters can be set for mixed pallet module:

workmode

- Online: Plan the stacking for each new box without knowing the size.
- Offline: Plan all the boxes when the size for all boxes to be stacked are known. This function is used to adjust the stacking parameters during the debugging phase. You may read the json

file to view the planned stacking. It does not support connecting and running physical robots.

- fileName: Read box json file name in Offline mode.

captureTwiceToUpdate Default False, it can be choose when you can’ t get the full size at the first photo, and you need to take a second photo because of the lack of the height information of the box.

When *visual_move* has grasped the box but has no box height information, mixed pallet will estimate the box height and plan according to the estimated height when calculating the placement position and weight;

When running this module to take a second photo to get the height of the box, the software makes a second planning. At this time, the mixed pallet will uses the box size given by two visual recognition to calculate.

Pallet Type Pallet common parameters:

PalletDimension

- palletSizeX: Set pallet length
- palletSizeY: Set pallet weight
- palletMaxZ：Set max pallet height

FixedSequence

- isFixedSequence：Default False, it can be choose when the order of boxes is fixed.
- futureGainFactor：This parameter takes effect when :guilabel:` isFixedSequence` is checked. If this parameter is increased, more consideration will be given to the subsequent box stackability when palletizing, and more space will be reserved.

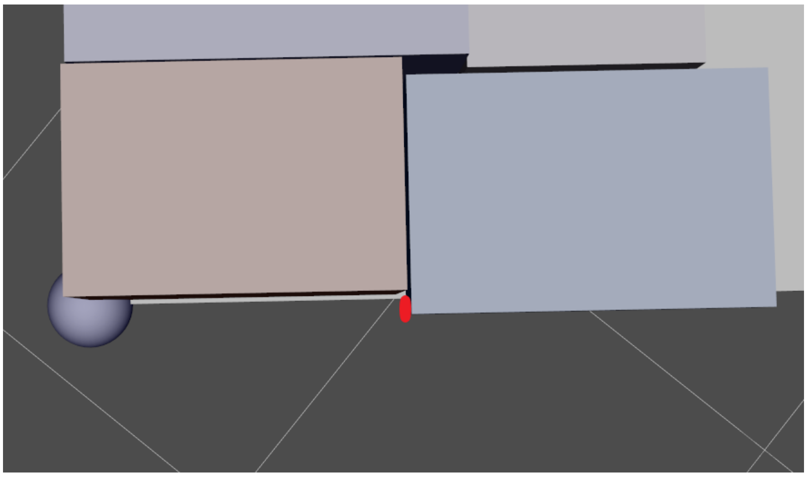
gapWidth The width of the gap between boxes, the unit is meter (m). To ensure that the given box size is smaller than the actual size to prevent collisions. Recommended setting: 0.01 – 0.02m

TaggedBox

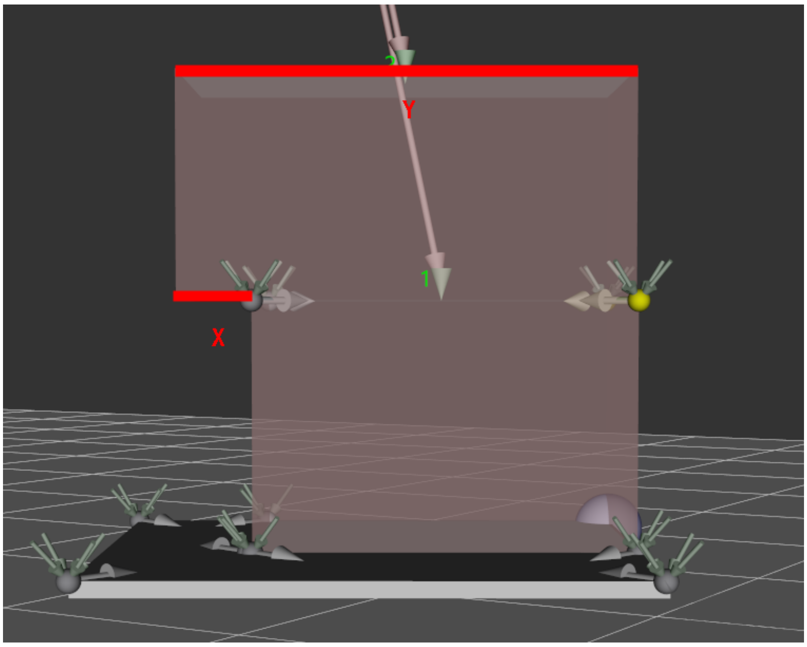
- Tag on Boundary：Default False.it can be choose when the boxes have the tag and the tag is on boundary.
- tagToBoundaryDist：The farthest distance from the box tag face to the edge of the pallet when stacking.

Structured

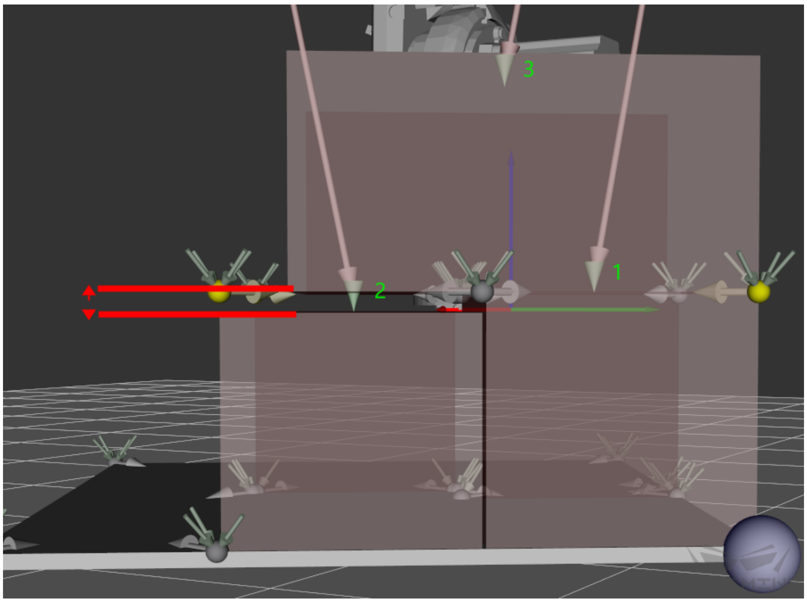
- Setting
  - allowedExcess: Allow the box to exceed the width of the edges. The recommended setting is 0.02 – 0.05m, as shown below:



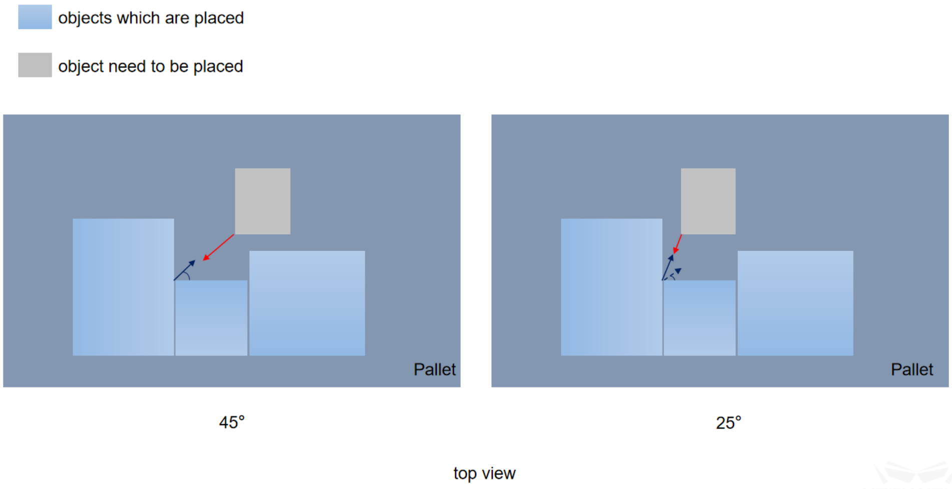
- boxExceedRatio: The upper box is allowed to exceed the maximum proportion of the plane on which it is pressed. As shown in the figure, the maximum Y/X does not exceed this parameter.



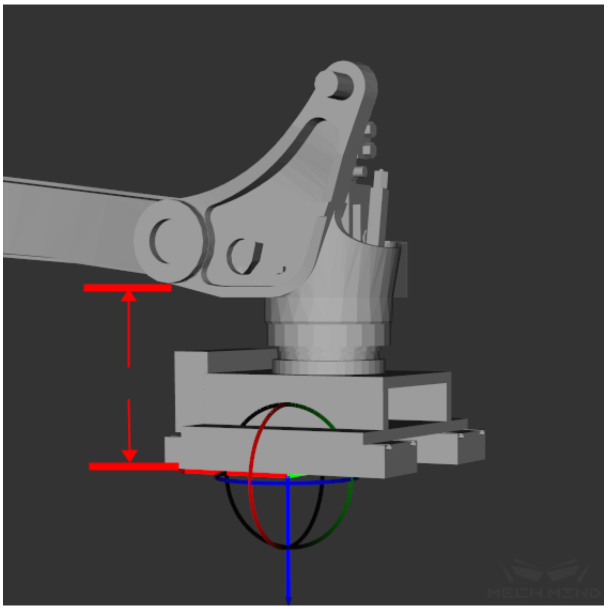
- layerHeightDiff: When the size of the box to be stacked on the upper layer is larger than the box on the lower layer, the large box on the upper layer is allowed to be stacked on a plane whose height difference does not exceed this parameter.



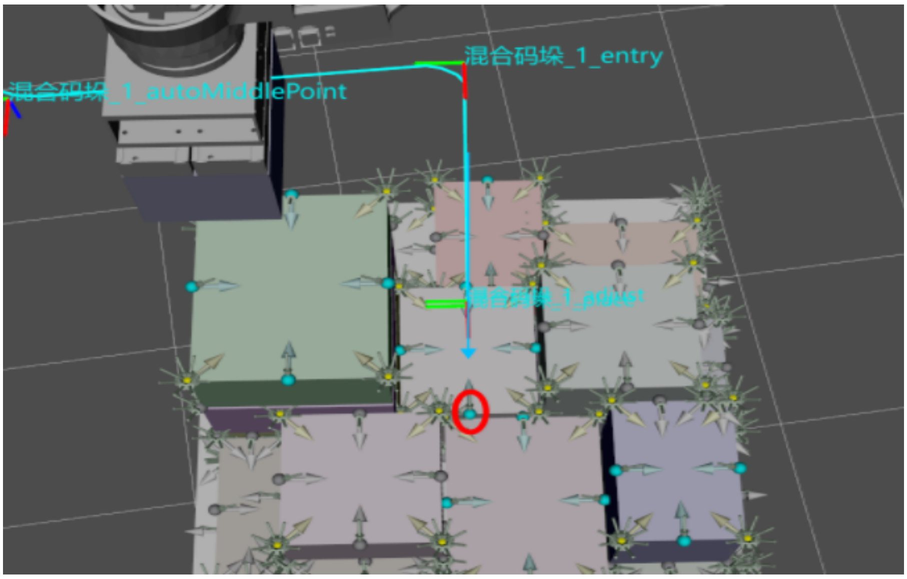
- samplingRate: Plan the sampling rate (sample/meter) for the box position. The higher the sampling rate, the more accurate the result, but the slower the speed. Recommended values are 200, 500, and 1000 sample/m.
- cornerFreeAngle: In the projection direction of the pallet, the angle between the entry path of the box and the side of the adjacent box. As shown in the figure, if this parameter is set too large, a U-shaped empty area may be left during stacking; if it is set too small, the actual stacking boxes may collide with adjacent boxes. It is recommended to set 15 – 30 degrees.



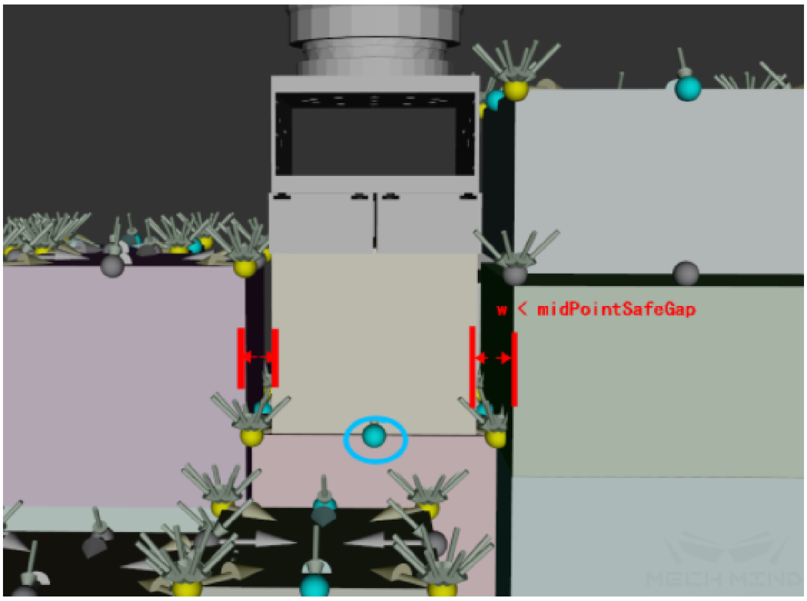
- endEffectorDimZ: The vertical distance from the bottom surface of the robot’ s end to the bottom surface of the robot’ s penultimate joint, as shown in the figure below, to prevent some placement poses from causing the end to collide with the robot body.



- enableMidPoint: The default is True. In addition to the corner points will be as candidate positions, the midpoint between the corner points will also be used as candidate positions. When there are U-shaped grooves in the stack, the boxes may be inserted into the grooves in a trajectory parallel to the adjacent boxes. The middle point is displayed in cyan. As shown in the figure



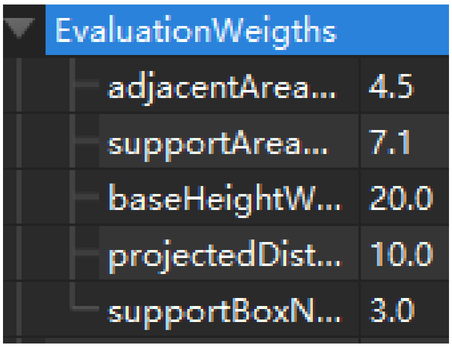
- midPointSafeGap: This parameter will be effective when setting Use Midpoint to True. When the box is inserted at the midpoint candidate position, the gap left on both sides will be larger than this value. As shown in the figure below.



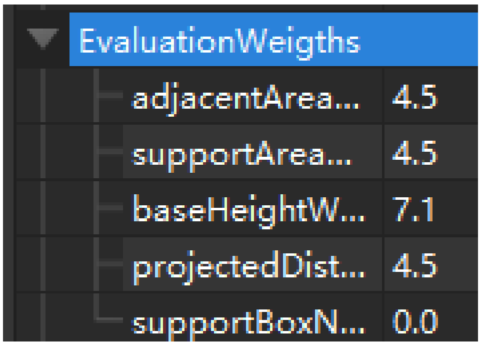
If the robot motion error and the box size error allow, this parameter can be set to a value smaller than *gapWidth*, which will significantly improve the pallet pattern when workMode is online.

- Weight：The parameters in this group are weights, which decides the box’s place position in the pallet.
  - adjacentAreaWeight: The higher the value of this parameter, the more likely that the candidate position where the side touches the box in the stack will be executed, and vice versa.
  - supportAreaWeight: The higher the value of this parameter, the larger the supported area (area which is less exceed the lower surface area) is more likely to be executed, and vice versa.
  - baseHeightWeight: The higher the value of this parameter, the lower the target position is more likely to be executed, and vice versa.
  - projectedDistToCornerWeight: The higher the value of this parameter, the shorter the distance from the position to the priority corner is projected on the diagonal of the pallet, which is easier to be performed.
  - supportBoxNumWeight: The higher the value of this parameter, the position above where more boxes have been stacked is more likely to be executed, and vice versa. The higher the value makes it easier to plan a relatively more stable stack with reducing the compactness of the stack.

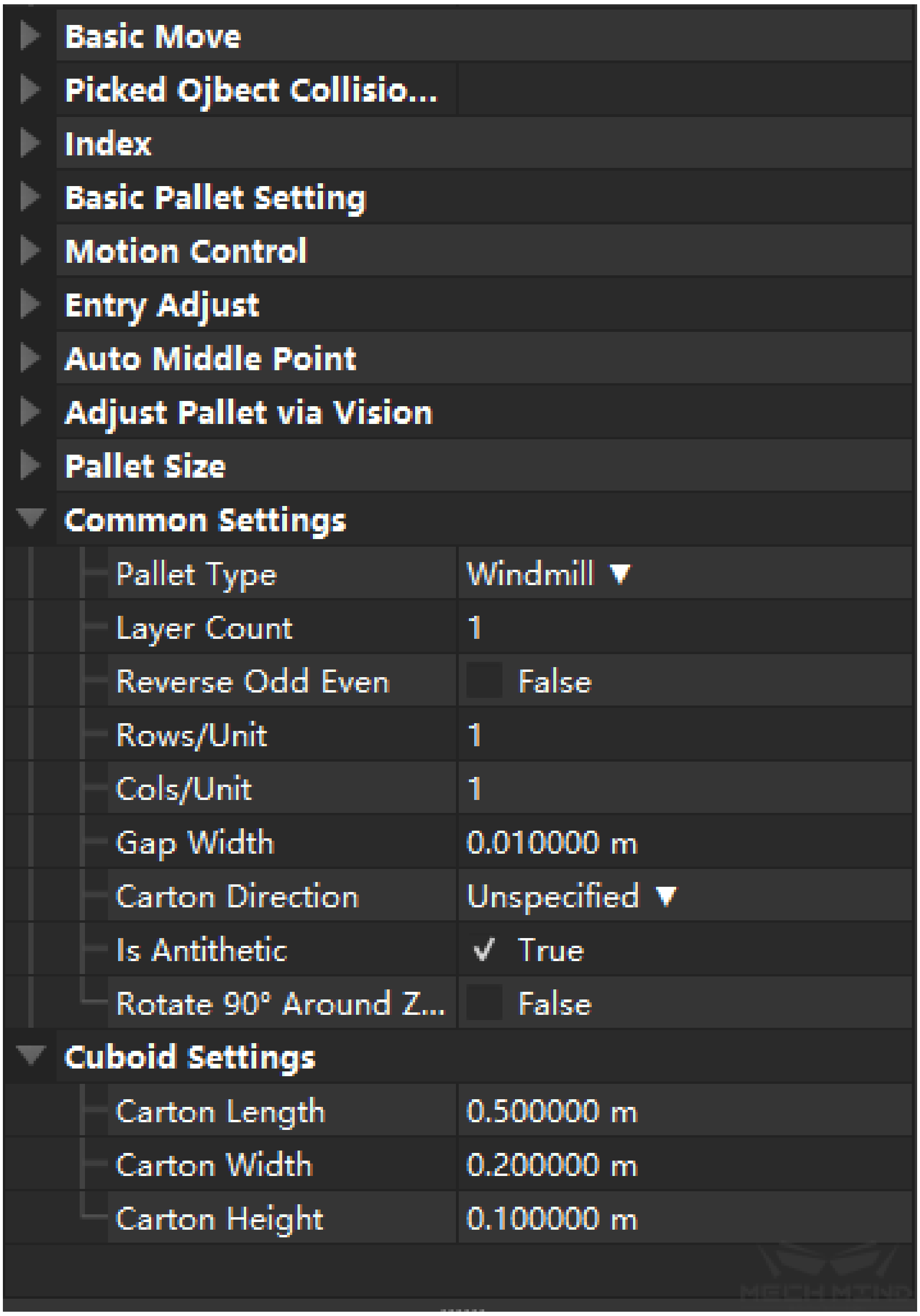
  It is recommended to use multiples as the adjustment interval during initial adjustment. Recommended parameters:



EvaluationWeigths	
adjacentArea...	4.5
supportArea...	7.1
baseHeightW...	20.0
projectedDist...	10.0
supportBoxN...	3.0



EvaluationWeigths	
adjacentArea...	4.5
supportArea...	4.5
baseHeightW...	7.1
projectedDist...	4.5
supportBoxN...	0.0



- Basic Move
- Picked Ojbect Collisio...
- Index
- Basic Pallet Setting
- Motion Control
- Entry Adjust
- Auto Middle Point
- Adjust Pallet via Vision
- Pallet Size
- Common Settings

Pallet Type	Windmill ▼
Layer Count	1
Reverse Odd Even	□ False
Rows/Unit	1
Cols/Unit	1
Gap Width	0.010000 m
Carton Direction	Unspecified ▼
Is Antithetic	✓ True
Rotate 90° Around Z...	□ False

- Cuboid Settings

Carton Length	0.500000 m
Carton Width	0.200000 m
Carton Height	0.100000 m

Unstructured

- Weight：The parameters in this group are weights, which decides the box’ s place position in the pallet.
  - heightWeight：The higher this parameter, the more consideration should be given to the height of the stack type. The height of the stack type should not be too high.
  - capacityWeight：The lower this parameter, the more compact the stack.
  - roughnessWeight：The higher this parameter, the smoother the top layer of the stack.
  - areaWeight：When stacking from boxes, the lower this parameter, the boxes with

larger projected area will be selected first.

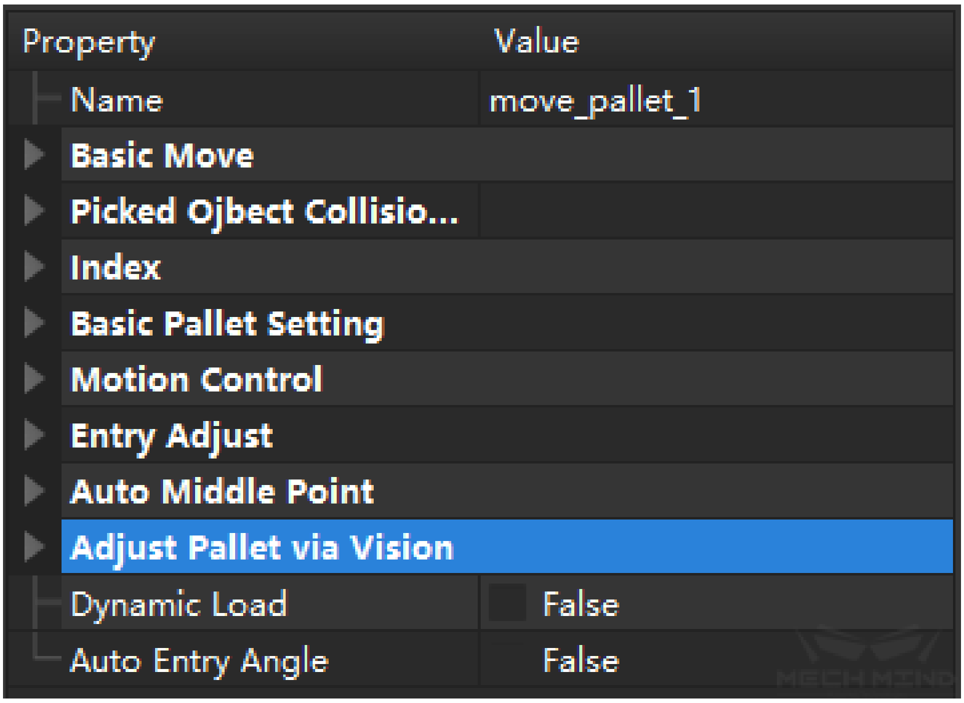
- distanceToBoundaryWeight: The higher this parameter, the closer the box to the pallet boundary.
- baseHeightWeight: In multi-layer stacking, the lower this parameter, the easier it is to stack the boxes on the lower plane

4.11.4 custom_pallet_pattern

Description

Using custom_pallet_pattern module, you can define any pallet pattern and is available at any time, but correspondingly, the use of it is relatively complicated.

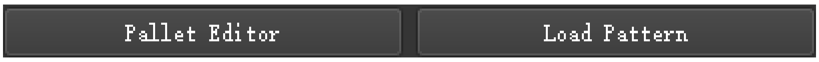
Parameters



Property	Value
Name	move_pallet_1
▶ Basic Move	
▶ Picked Ojbect Collisio...	
▶ Index	
▶ Basic Pallet Setting	
▶ Motion Control	
▶ Entry Adjust	
▶ Auto Middle Point	
▶ Adjust Pallet via Vision	
Dynamic Load	□ False
Auto Entry Angle	False

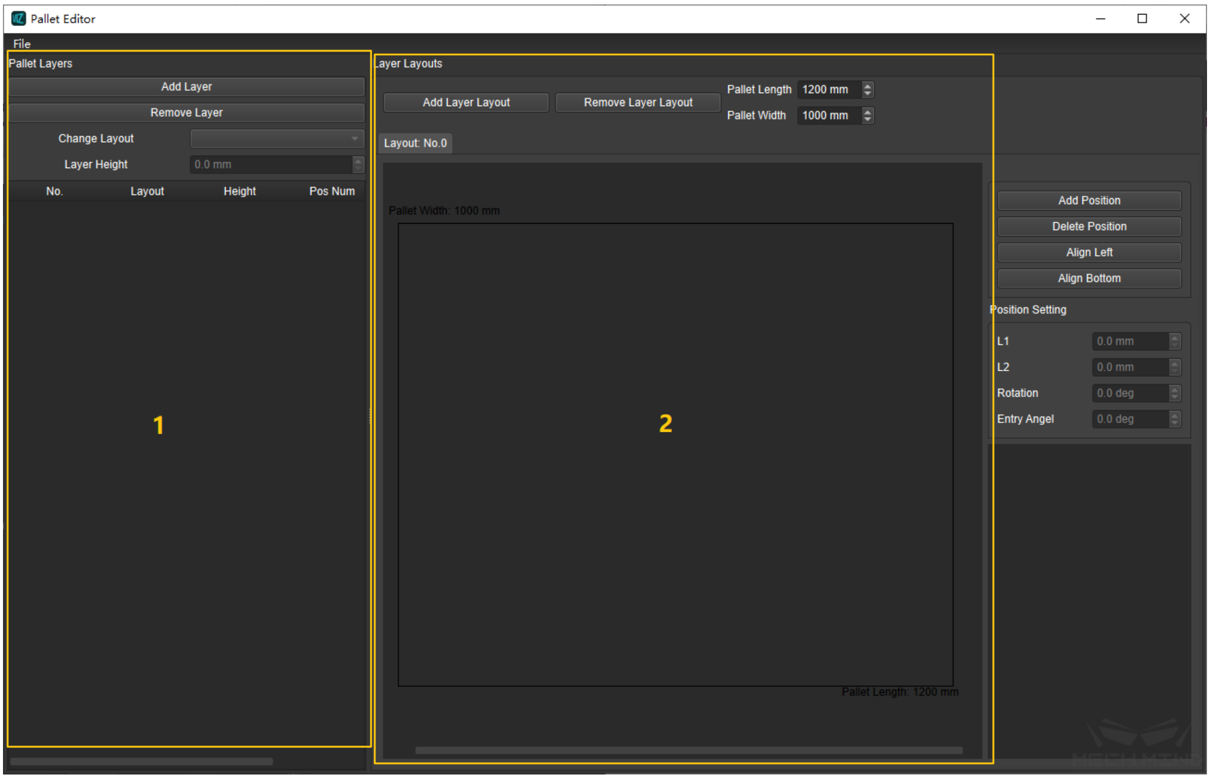
palletEdit

In addition to *Parameters* , custom_pallet_pattern has the functions of pallet edit and pattern loading



Click *Pallet Editor* to pop up the interface of the pallet pattern editor. When the pallet pattern editing is finished, the pallet pattern will be saved in the project folder as a json file. Click *Load Pattern* to select the completed json file to load the edited pallet pattern.

The interface of Pallet Editor is shown below. The left and right sides are the overall editing area and layout editing area respectively.



Layer Height 0.0 mm

Pallet Length 1200 mm

Pallet Width 1000 mm

L1 0.0 mm

L2 0.0 mm

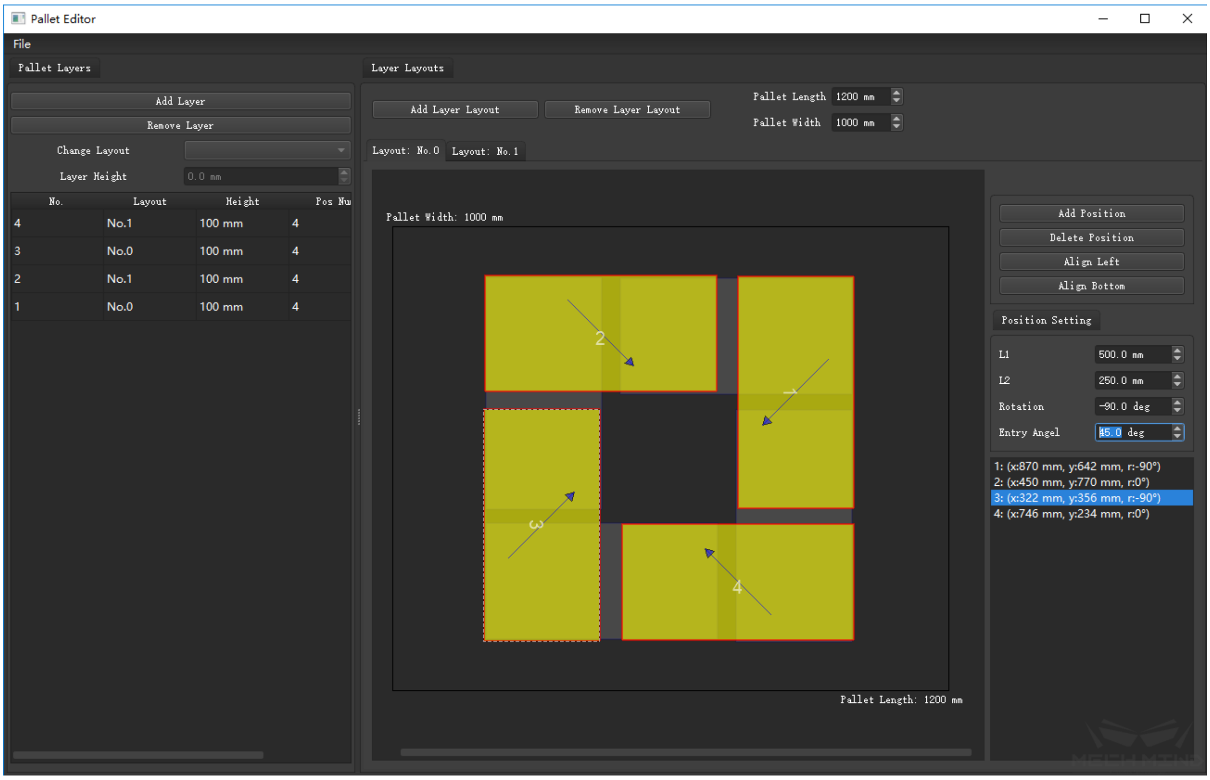
Rotation 0.0 deg

Entry Angel 0.0 deg

The number of stacking layers, the layout and height of each layer can be set in the overall editing area;

The number of boxes, the position adjustment of boxes, the specific layout editing, and the parameters such as box length and width, rotation, and plane cut-in angle editing can be set in the layout editing area.

An example of an edited pallet pattern:



Change Layout

Layer Height 0.0 mm

No.	Layout	Height	Pos Nu
4	No.1	100 mm	4
3	No.0	100 mm	4
2	No.1	100 mm	4
1	No.0	100 mm	4

Pallet Length 1200 mm

Pallet Width 1000 mm

L1 500.0 mm

L2 250.0 mm

Rotation -90.0 deg

Entry Angel 45.0 deg

1: (x:870 mm, y:642 mm, r:-90°)
2: (x:450 mm, y:770 mm, r:0°)
3: (x:322 mm, y:356 mm, r:-90°)
4: (x:746 mm, y:234 mm, r:0°)

As shown on the left side of the figure above, the whole stack has four layers, but there are only two layouts, wherein the layout layer No.0 is used for the odd-numbered layers and the layout No.1 is used for the even-numbered layers.

The right side of the figure is a top view of the pallet pattern, wherein each yellow/gray block represents a box, a yellow box represents a box in the current layout, and a gray box represents a box in another layout.

The operation is as follows:

- Pallet Layers Tab:
  - Click *Add Layer* to increase the number of layers to 4, click each layer one by one, and select a layout for each layer in the `Change Layout` menu.
- Layer Layouts Tab:
  - Click *Add Layer Layout* to add different layout modes.
  - In a certain layout mode, click *Add Position* to add a box. The box layout supports some common operations such as *Align Left* and *Align Bottom*.
  - When a single box is selected, you may change the length (L1), width (L2), rotation angle(Rotation), and cut-in angle(Entry Angle) of the box (that is, the box’ s entry angle on the plane) in Position Setting Tab.
  - After editing, you may click *File* → *Save to File* in the upper left corner and save it to an external json file for further use.

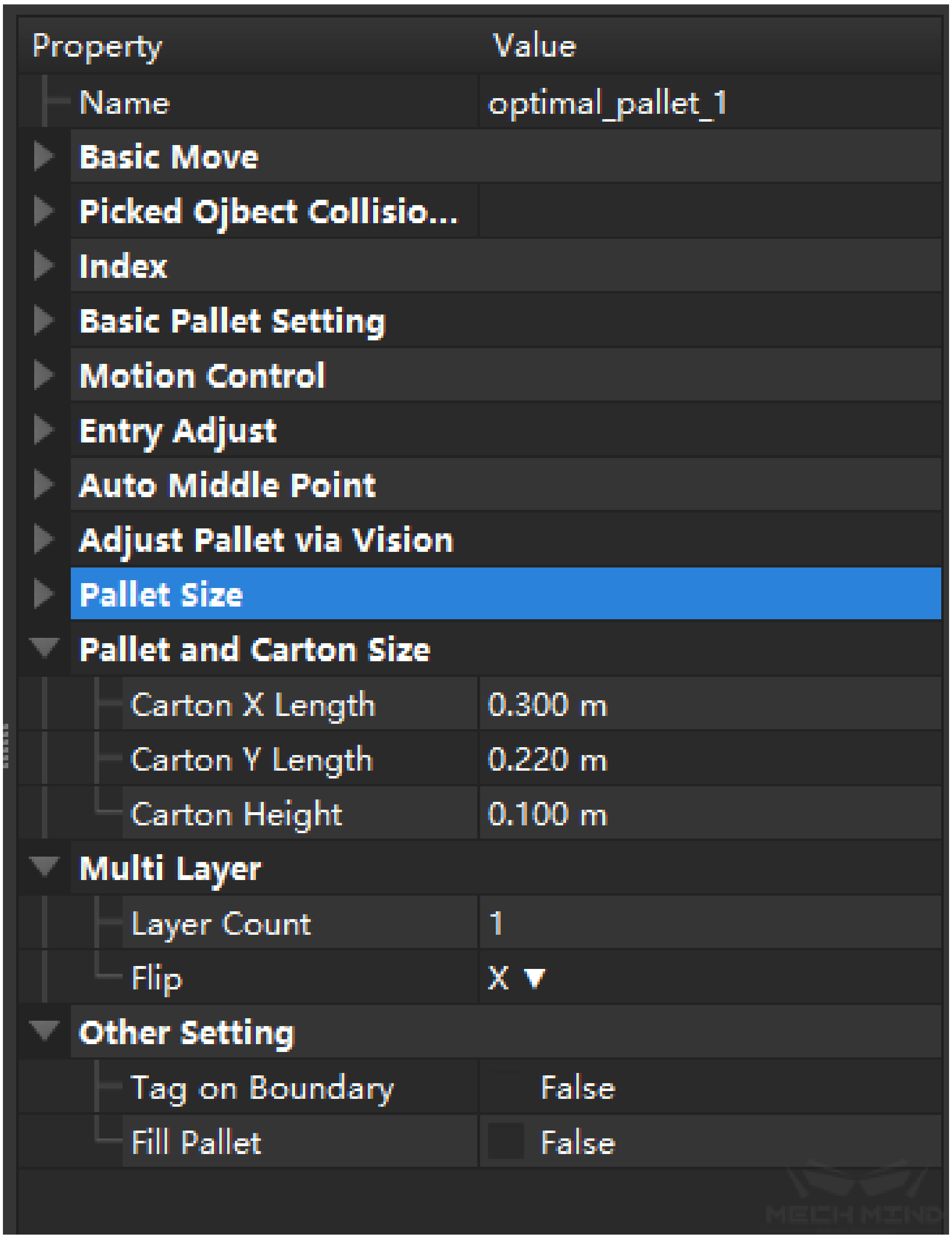
4.11.5 optimal_pallet

Description

optimal_pallet can automatically plan the pallet pattern with the largest number of stacked boxes based on the given box size and pallet size. In addition, when generating a pallet pattern, this function plans the stacking order in the specified entrance direction; stacking according to this order can prevent the boxes from the collisions during the stacking process.

Note: The stacking sequence generated by this module starts from the corner away from the automatic middle point(AutoMiddlePoint).

Parameters



Property	Value
Name	optimal_pallet_1
Basic Move	
Picked Ojbect Collisio...	
Index	
Basic Pallet Setting	
Motion Control	
Entry Adjust	
Auto Middle Point	
Adjust Pallet via Vision	
Pallet Size	
Pallet and Carton Size	
Carton X Length	0.300 m
Carton Y Length	0.220 m
Carton Height	0.100 m
Multi Layer	
Layer Count	1
Flip	X ▼
Other Setting	
Tag on Boundary	False
Fill Pallet	□ False

PalletAndCartonSize

- palletXLen, palletYLen: The length of the pallet X-side and Y-side under the pallet. The unit is meter;
- cartonXLen, cartonYLen, cartonHeight: The length of the box X-side, Y-side, and height of the box. The unit is meter

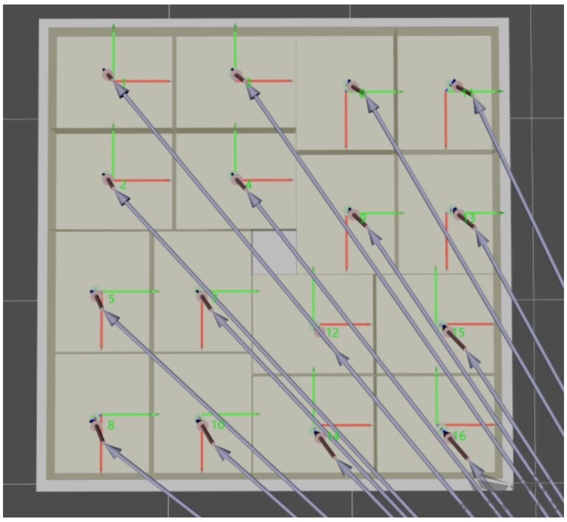
MultiLayer

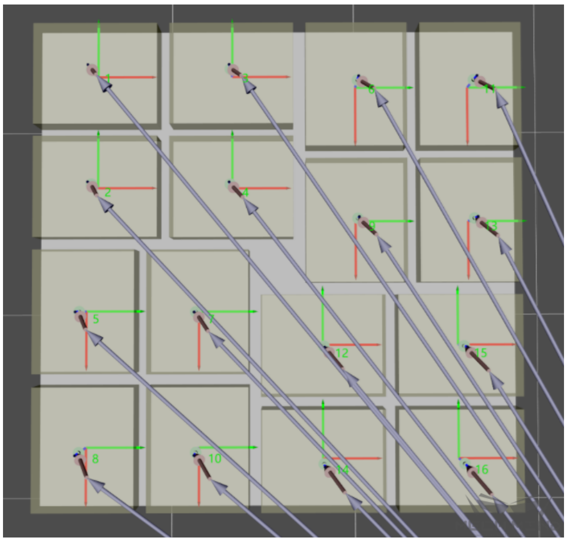
- layerCount: The total number of layers of the pallet pattern.

- flip: The interleaving manner of the adjacent interlayer.
  - X: The adjacent layer pallet pattern is obtained by mirroring the lower layer around the X axis of the pallet
  - Y: The adjacent layer pallet pattern is obtained by mirroring the lower layer around the Y axis of the pallet
  - Rotate: The adjacent layer pallet pattern is obtained by rotating the lower layer 90 ° around the geometric center of the pallet, and it is only used when the generated pallet pattern is square.

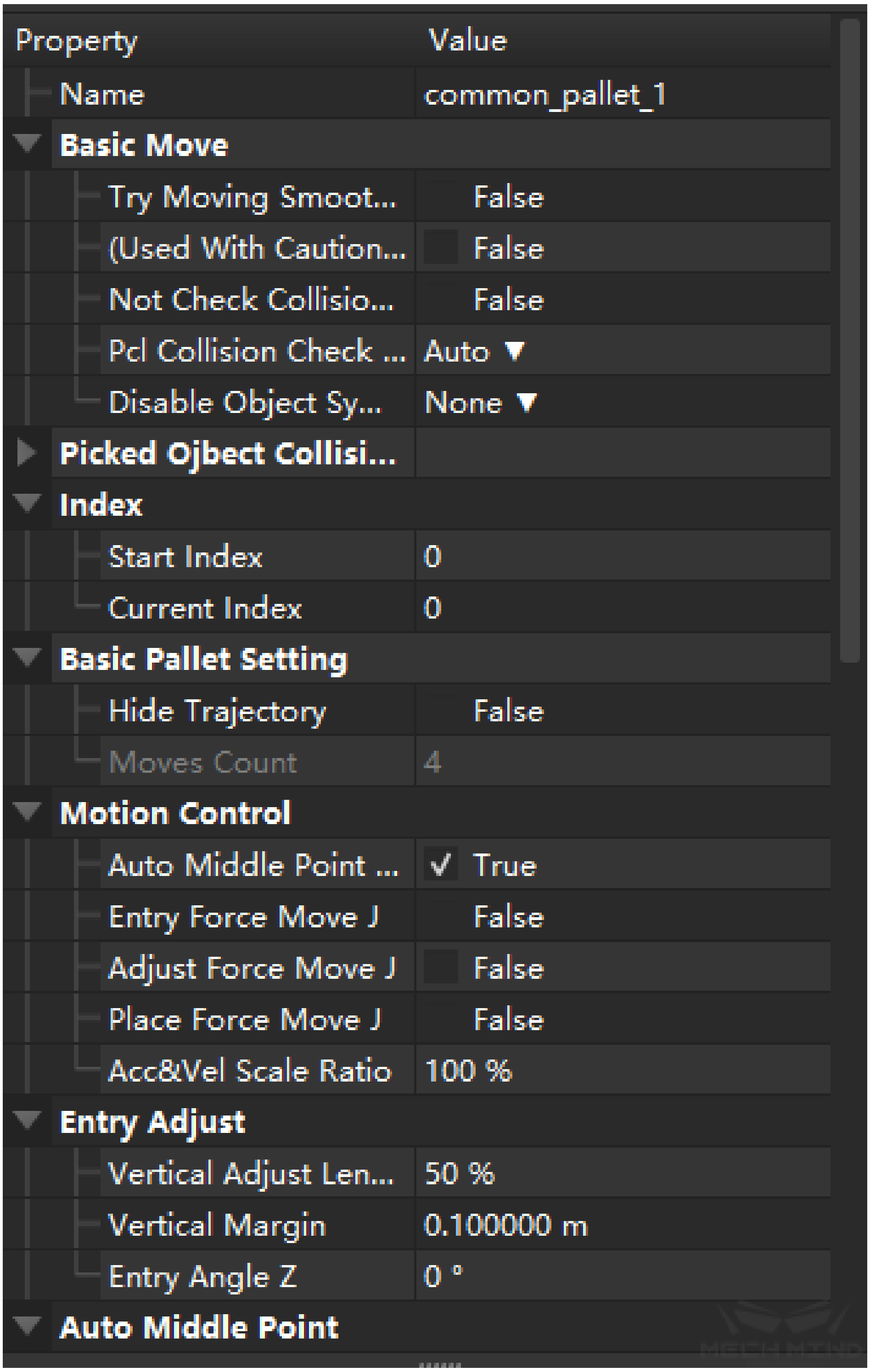
AdvancedSetting

- fillPallet: Whether the stack is evenly spread over the entire pallet. The default is False. The planned pallet pattern is the smallest pallet pattern that can be achieved, and the geometric center of the pallet pattern is translated to the geometric center of the pallet, as shown in the left. If True is selected, the boxes are evenly spread on the pallet, as shown on the right.





4.11.6 Parameters



Property	Value
Name	common_pallet_1
Basic Move	
Try Moving Smoot...	False
(Used With Caution...	False
Not Check Collisio...	False
Pcl Collision Check ...	Auto ▼
Disable Object Sy...	None ▼
Picked Ojbect Collisi...	
Index	
Start Index	0
Current Index	0
Basic Pallet Setting	
Hide Trajectory	False
Moves Count	4
Motion Control	
Auto Middle Point ...	✓ True
Entry Force Move J	False
Adjust Force Move J	False
Place Force Move J	False
Acc&Vel Scale Ratio	100 %
Entry Adjust	
Vertical Adjust Len...	50 %
Vertical Margin	0.100000 m
Entry Angle Z	0 °
Auto Middle Point	

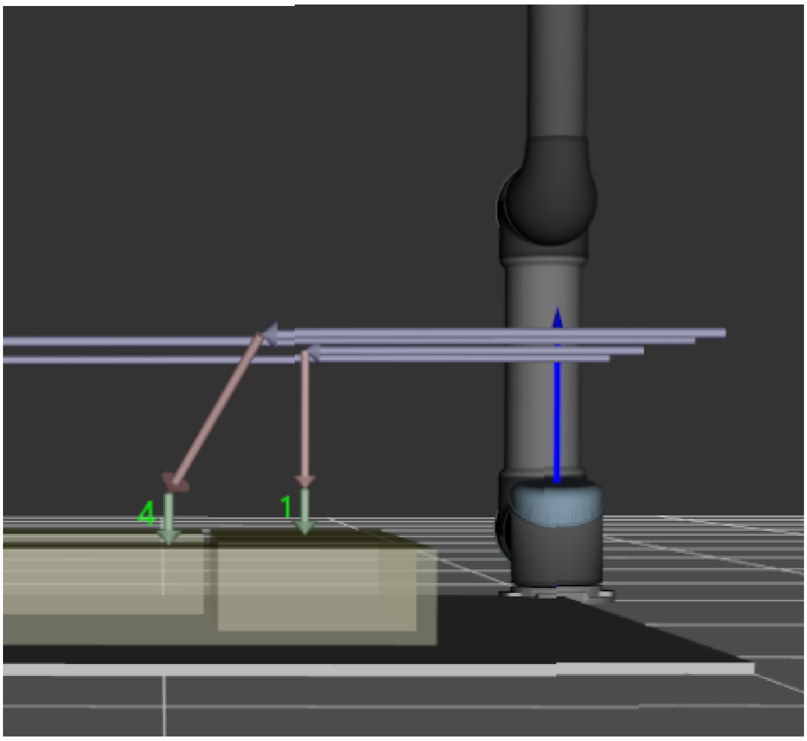
Index The meaning of this parameter is to index the box to be placed, counting starts from 0. The application method is: when starting the code from an empty stack, this value starts from 1. If four boxes have already been stacked, the stacking process is suspended for some reasons at this time. When the stacking is started again, it is not necessary to manually remove the boxes that have been stacked on the pallet and start again from the beginning. Instead, you only need to set the start index to 4. The program can automatically continue to execute the palletizing process from the fifth box.

BasicPalletSetting

- hideTrajectory
- movesCount

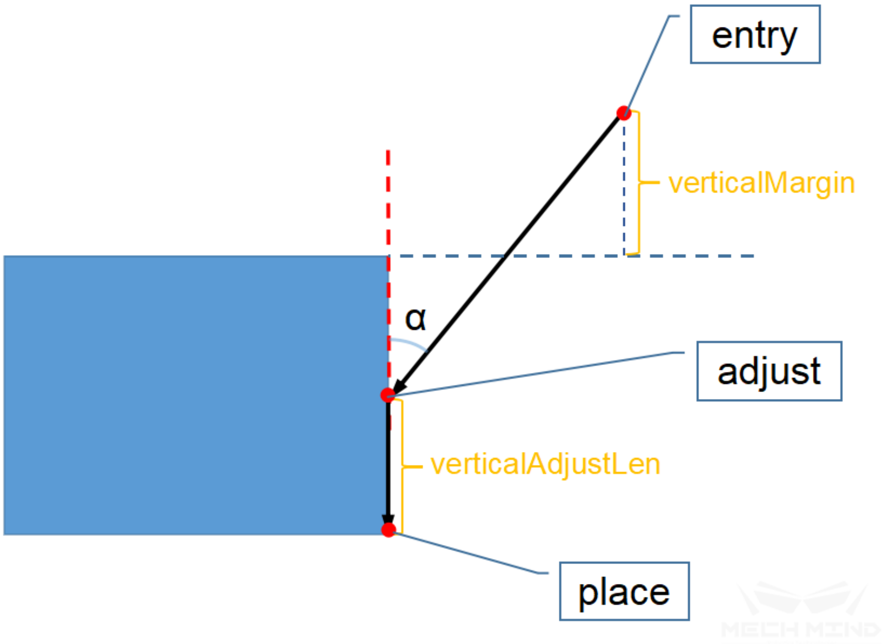
MotionControl

- autoMiddlePointForceMoveJ
- entryForceMoveJ
- adjustForceMoveJ
- placeForceMoveJ
- accVelScaleRatio: As shown in the figure below, when approaching the stack, it is divided into three stage. The first stage approaching the stack is in purple, and the other two stage are the paths for actually placing the boxes. According to the field application, sometimes the robot is required to move faster when approaching the stack, and slower when actually placing the boxes, in such a case, the speed/acceleration of the two stages of approaching the stack (purple) and actually placing the boxes can be restricted by this parameter. The speed and acceleration approaching the stack (purple) are the values specified in *Basic Move General Parameters*, referred to as v1, and the speed/acceleration of the next two stages of robot motion are v1 × this parameter.



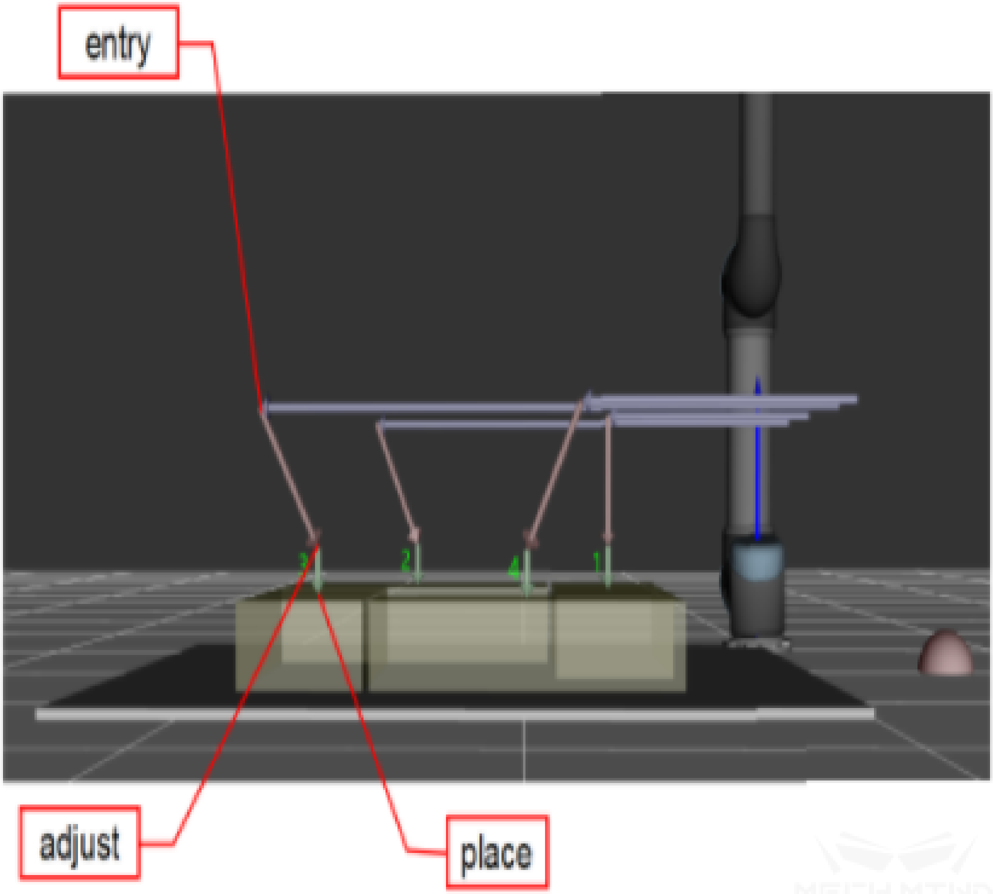
EntryAdjust There are three parameters in this set, which control the entry path of boxes together when they are stacked. The purpose is to avoid the collision of the stacked boxes due to accuracy or other reasons when the robot directly lowers the boxes vertically. Adjust the entry path so that the box approaches the stacked box at a certain angle and then lowers it vertically;

For each box, there are 4 positions in the stack. The three parameters in this set control the three parameters, as shown by the red dots in the figure below, which are entry, adjust and place. The perspective below is the front view of the box.

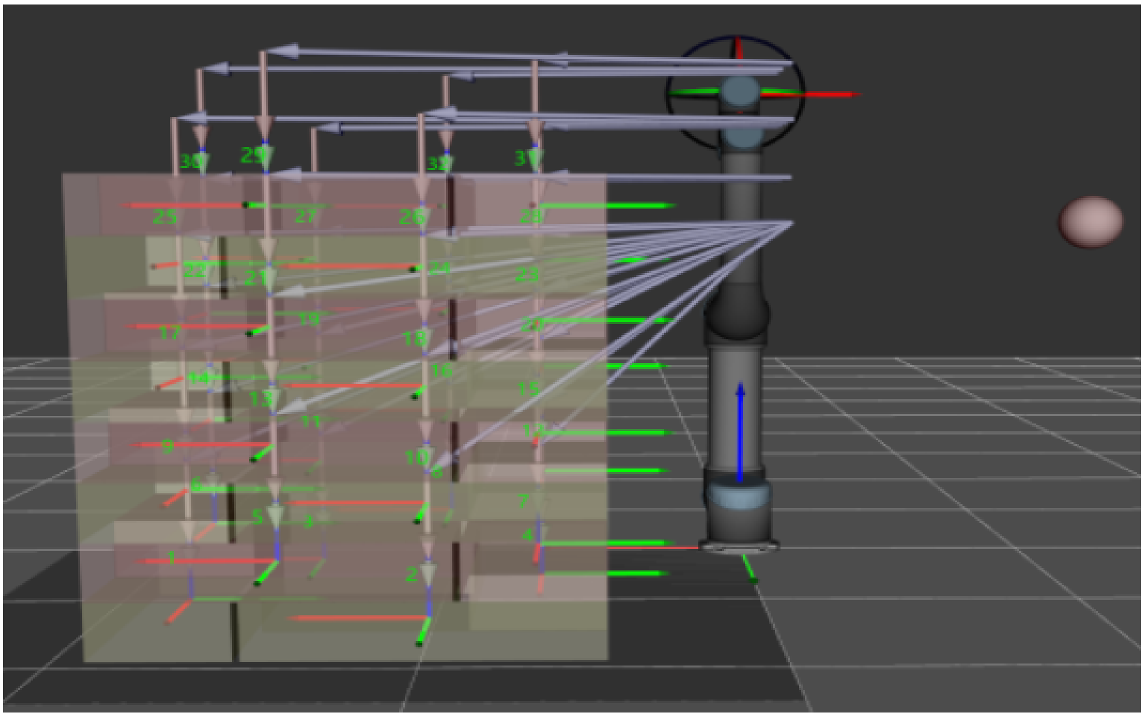


- verticalAdjustLenRatio: This parameter is the height ratio when approaching the stacked box, which is equal to (verticalAdjustLen / box height). The value range is 0 – 1, with no unit. The default value is 0.5.
- verticalMargin: This parameter is the height margin of the cut-in point. The value ranges from 0 to infinity, and the unit is millimeter. Generally, it is set to a number greater than 0 to leave a margin. The specific value depends on the application scene.
- entryAngleZ: This parameter is in the above figure. It is the angle between the path entry-adjust and the vertical direction. The unit is °, and the value range is -80 – 80 °. The recommended default value is 30 ° – 45 °.

autoMiddlePoint: The pink ball as shown in the figure below, this set of parameters controls the fourth position in the stack is mentioned above-the position of the middle point.



- X/Y: Set the position x, y of the pink ball in the base coordinate system of the robot. The software will automatically calculate reasonable midpoint coordinates for stacks of different heights based on this position.
- minZ: The minimum absolute z-height when approaching (purple path). (And the maximum height of this floor). As shown below



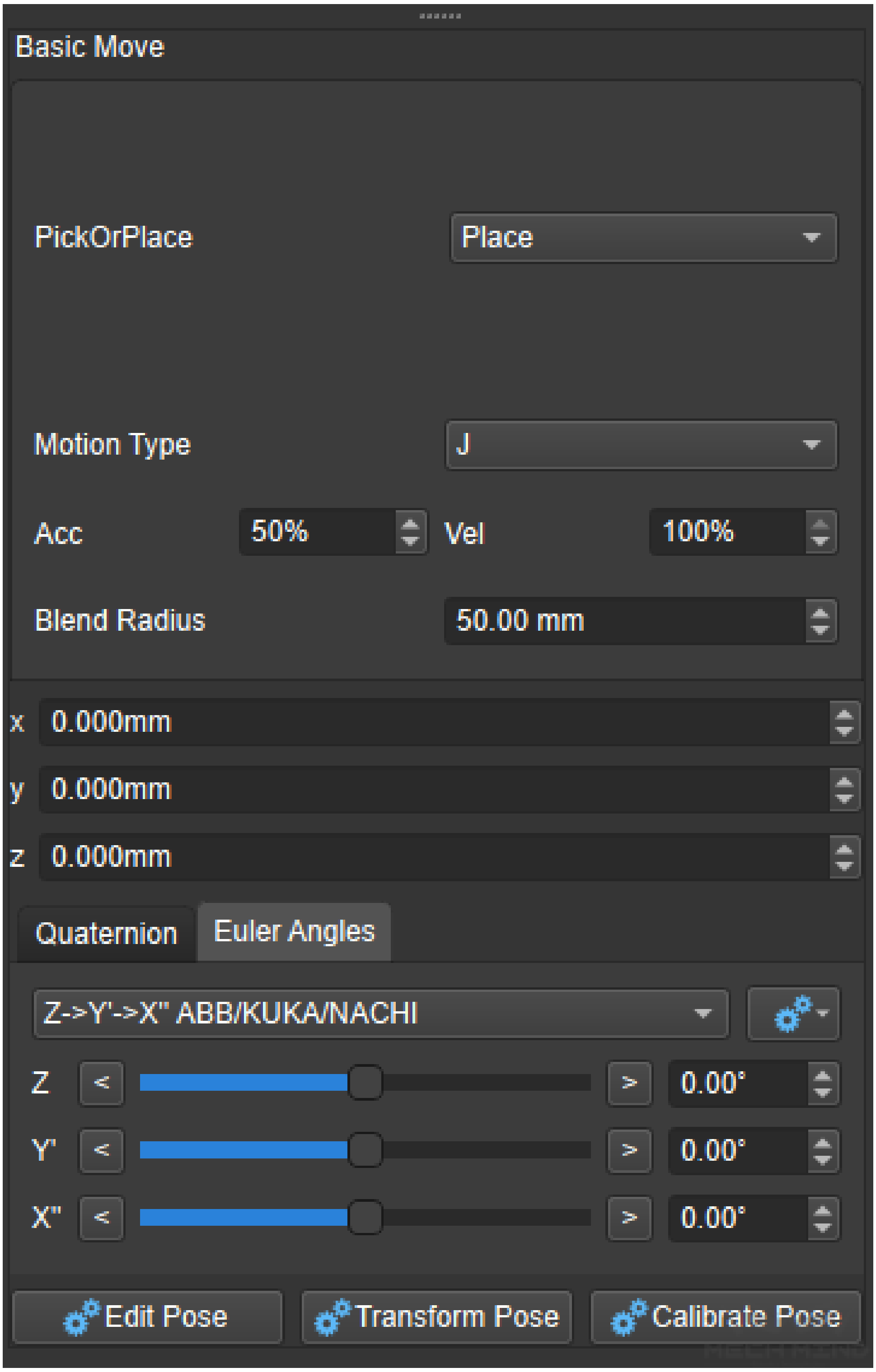
- isMiddlePointTrajVertical
- extendedDist

> **Attention:** The autoMiddlePoint is only an indication of the direction when approaching the stack, not the point that the robot actually arrives. Therefore, the pink ball shall be as far away from the pallet as possible. If the ball is too close to the stack, during the process of placing the box, it may cause crushing collision.

AdjustPalletViaVision This parameter can dynamically adjust the position of the stack by the vision service

- needAdjustPalletPose: The default is False. When the position of the stack is required to be adjusted dynamically, select it as true. When the program proceeds to this task, it will call the vision service to identify the location of the stack.
- visionToAdjustPallet: Fill in the name of the vision project to identify the location of the stack. When the program proceeds to this task, it will call the vision service according to the vision project name.

BasicMove



Basic Move

PickOrPlace Place

Motion Type J

Acc 50% Vel 100%

Blend Radius 50.00 mm

x 0.000mm

y 0.000mm

z 0.000mm

Z->Y'->X" ABB/KUKA/NACHI

Z 0.00°

Y' 0.00°

X" 0.00°

Here you can set the position of pallet and move parameters, etc. The function is the same as other *Basic Move General Parameters* , so it won’ t be described here.

4.12 Trajectory

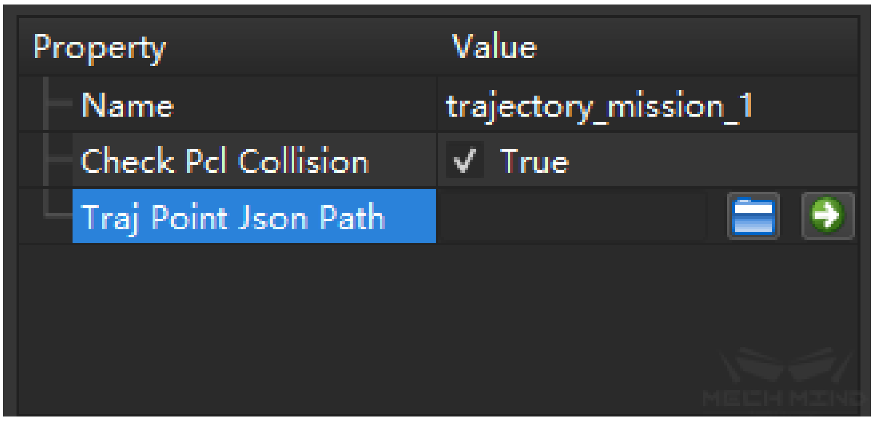
The module in this group is applied to gluing.

4.12.1 trajectory_mission

Description

Perform multiple continuous dense point movements according to the template file of trajectory point. It need to be used with *visual_move* .

Parameters



Property	Value
Name	trajectory_mission_1
Check Pcl Collision	✓ True
Traj Point Json Path	

trajPointJsonPath Pre-edited trajectory point json file. When the program is running, the number of points given by the vision service must be consistent with the number of points in the template.

Recommanded solution of moving continuous trajectory

If all the gluing trajectory need to be moved at once without interrupt, it is recommanded to check the *useAll* in *visual_move* .

Note: The number of trajectory points provided by the vision service in this way does not need to be fixed, and it is relatively simple to use.

CHAPTER

FIVE

FAQ

1. Can the scene model be imported?

   Yes. For details, please see *Scene*.

2. Speed of Robot:

   Vs = A * B * C

   Vs: The speed of robot for single step

   A: Automatic running speed of robot (Speed setting on the robot’ s teach pendant)

   B: Speed set in Mech-Viz

   C: Speed of single step of movement task in Mech-Viz

   For some robots, A may not exist or it not be effective, please see the manuals of the used robot for details.