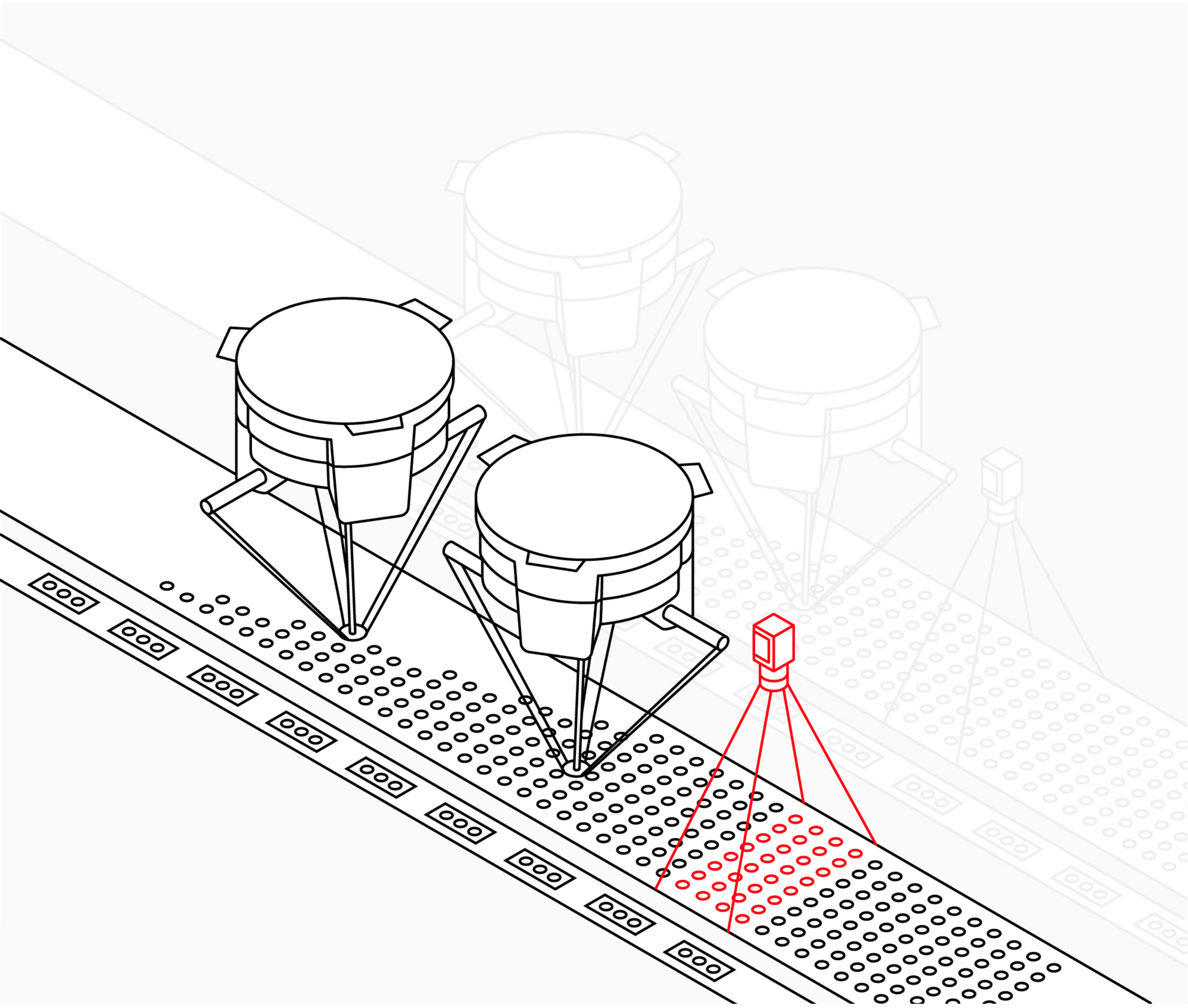
ROBOTICS

Operating Manual

PickMaster Twin Recipe Manager



Trace back information:
Workspace Main version a676
Checked in 2025-06-08
Skribenta version 5.6.019

Operating Manual

PickMaster Recipe Manager Release 3.0.1

OmniCore and IRC5

Document ID: 3HAC092763-001

Revision: B

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damage to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

This page is intentionally left blank

Overview of this manual

About this manual

This manual describes how to use PickMaster Recipe Manager to modify items, containers and recipes. It also explains PickMaster Recipe Manager terms and concepts.

Usage

This manual should be used when working with PickMaster Recipe Manager.

Who should read this manual?

This manual is intended for PickMaster Twin users, proposal engineers, mechanical designers, offline programmers, robot technicians and service technicians.

Prerequisites

The reader should have basic knowledge of:

- Industrial robots and their terminology
- PickMaster Twin

Cybersecurity

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide, and continuously ensure, a secure connection between the product and to your network or any other network (as the case may be).

You shall establish and maintain any appropriate measures (such as, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its entities are not liable for damage and/or loss related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Note

Only qualified personnel should write or modify the script files.

It is the responsibility of the writer to make sure that the cell is safe when running with the script files.

Application ports and protocol types

The following table lists the used port numbers between the PickMaster Recipe Manager and other components, their communication protocol types and usage.

Port	Protocol Type	Usage
50000	TCP	RIS2 commands of PickMaster Runtime
6001	TCP	Emulation of PickMaster Runtime

Continues on next page

Port	Protocol Type	Usage
3	TCP	Vision client
5	TCP	Vision server
9000	TCP	Zenon event port
319-320	UDP	Time sync service
502	TCP	Modbus
34964	UDP	Profinet
44818	TCP	EtherNet/IP
2222	UDP	EtherNet/IP

Organization of chapters

The manual is organized in the following chapters:

Chapter	Content
1 Introduction	Describes terms and concepts of PickMaster Recipe Manager.
2 Getting started	Describes how to start PickMaster Recipe Manager.
3 Navigating PickMaster Recipe Manager	Describes the graphical user interface of PickMaster Recipe Manager.
4 Workflow for PickMaster Recipe Manager	Describes how to work with PickMaster Recipe Manager.

References

OmniCore

Reference	Document ID
Product specification - PickMaster® Twin	*3HAC092765-001*
Circuit diagram - PickMaster Twin	*3HAC024480-020*
Safety manual for robot - Manipulator and IRC5 or OmniCore controller [i]	*3HAC031045-001*
Application manual - Conveyor tracking	*3HAC066561-001*
Product manual - OmniCore C30	*3HAC060860-001*
Product manual - OmniCore C90XT	*3HAC073706-001*
Operating manual - OmniCore	*3HAC065036-001*
Operating manual - Integrator's guide OmniCore	*3HAC065037-001*
Application manual - Controller software OmniCore	*3HAC066554-001*
Technical reference manual - Event logs for RobotWare 7	*3HAC042927-001*
Technical reference manual - Lubrication in gearboxes	*3HAC042927-001*
Technical reference manual - System parameters	*3HAC065041-001*

[i] This manual contains all safety instructions from the product manuals for the manipulators and the controllers.

Continues on next page

IRC5

Reference	Document ID
Product specification - PickMaster® Twin	*3HAC073650-001*
Circuit diagram - PickMaster Twin	*3HAC024480-020*
Operating manual - RobotStudio	*3HAC032104-001*
Application manual - Conveyor tracking	*3HAC066561-001*
Product manual - IRC5	*3HAC047136-001*
Product manual - IRC5 Panel Mounted Controller	*3HAC027707-001*
Operating manual - OmniCore	*3HAC065036-001*
Operating manual - IRC5 Integrator's guide	*3HAC050940-001*
Technical reference manual - Event logs for RobotWare 7	*3HAC066553-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC065038-001*
Technical reference manual - RAPID Overview	*3HAC065040-001*
Technical reference manual - System parameters	*3HAC065041-001*

Revisions

Revision	Description
A	First edition.
B	Released with PickMaster® Twin 3.0.1. • Minor corrections. • Supported multi-language documentation.

Safety

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement.

Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual *Safety manual for robot - Manipulator and IRC5 or OmniCore controller*.

When using PickMaster® Twin products

- When using with PickMaster® Twin products, it is the user's responsibility to adhere to the relevant standards and safety directives. In addition, the application manuals for proper use must be observed.
- Only personnel with appropriate training and required knowledge are allowed to use PickMaster® Twin products.
- The integrator installing the PickMaster® Twin is responsible for the safety.
- Wherever possible, the auto mode of operation shall be performed with all persons outside the safeguarded space.
- An emergency stop must also be available to make sure the emergency stop function is enabled.
- PickMaster® Twin only provides Operational Stop (Program Stop). The integrator shall make sure that proper Normal Stop (machinery stop) is configured correctly in the system.
- Make sure the hazardous situation that resulted in the emergency stop condition no longer exists. Release the emergency stop button manually to remove the emergency stop condition.
- Stops for the machine is the responsibility of the integrator and must be addressed according to local legislation.
- The integrator is responsible to conduct a risk assessment of the final application.
- Sensitive body parts, such as the eyes and the larynx, must be protected by personal protective equipment (PPE).
- Protective measures should be the precondition when using PickMaster® Twin products. PickMaster® Twin does not guarantee the robot targets are always in safe zone. It is integrator's responsibility to take protection measures, like using safe-move or setting proper robot work range etc.

Continues on next page

- Safety related status and operations shall be handled on the controller and by safety rated systems. PickMaster® Twin status information shall not be used as input for safety related information and operations.
- Protective measures should be the precondition when install/adjust/replace hardware parts, for example, the camera.
- The stop functions in PickMaster® Twin can never be used to replace A-stop/E-stop or any other safety related stops.

Note

If PickMaster Twin obtains the status of active AS/ES from the robot controller, the items in the queue will not be cleared. Once the AS/ES is deactivated, the users can resume the production by clicking the **Start**/**Unhold**/**Unsuspend** button on PickMaster Twin.

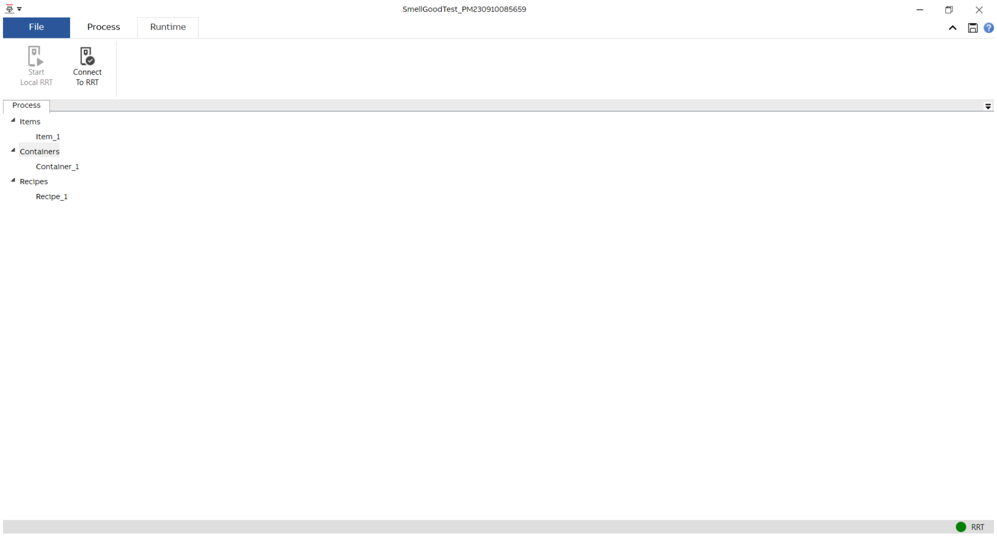
This page is intentionally left blank

1 Introduction

1.1 About PickMaster Recipe Manager

Overview

PickMaster Recipe Manager is a specific add-in to PickMaster Operator. This software product aims to load an existing solution to create, edit or delete items, containers, recipes and vision models without RobotStudio and PickMaster Powerpac.



xx2300002056

1.2 PickMaster Recipe Manager terms

About these terms

Some words have a specific meaning when used in this manual. Definitions of these words in this manual are listed below. Some of the terms are put in their context when describing a picking and placing process.

Term list

Words that have italic font style in the definition column are included in the term list and have their own definitions.

Term	Definition
PickMaster PowerPac	The market name of PickMaster PC engineering software that is used for simulating and commissioning picking lines with virtual and real Runtime.
PickMaster Operator	The market name of PickMaster production operator interface software that is used for running PickMaster applications in production. PickMaster Operator can read and write to solutions generated by PickMaster Recipe Manager. It has access to real Runtime.
PickMaster Recipe Manager	The market name of the software for creating, modifying items, containers and recipes on the host computer in the factory installation. The Recipe Manager is launched from PickMaster Operator.
PickMaster Virtual Runtime (VRT)	The core engine that orchestrates all the calculation of virtual pick and place operation in simulations.
PickMaster Real Runtime (RRT)	The core engine that orchestrates all the calculation of pick and place operation in real product. Runtime communicates with cameras and the robot controllers. It's also called as Runtime.
PickMaster Twin Client	It's the Client computer for configuring, simulating, and commissioning a PickMaster PowerPac solution. The PickMaster Twin Client installation package shall be installed on the Client computer. It contains PickMaster PowerPac, PickMaster virtual Runtime and PickMaster real Runtime.
PickMaster Twin Host	It's the Host computer for operating and managing PickMaster Twin in production. The PickMaster Twin Host installation package shall be installed on the Host computer. It contains PickMaster Operator and PickMaster real Runtime.
Solution	Format for storing a PickMaster Twin configuration result.
Recipe	Format and a collection of parameters regarding the process of Pick and Place for storing the process to be executed in a station.
Layout	Description of static objects in a PickMaster installation, for example robots, *work areas*.
Process	Description of a PickMaster picking process and all items, containers and recipes.
Work area	A defined picking and placing area for the robots.
Item	The generic term for a specific object to be picked or placed in a PickMaster Recipe Manager application.

Continues on next page

Term	Definition
Container	Defines a shape that can set specific patterns and what *items* to use for each position in the patterns.
Position generator	Defines the sensor configuration on the conveyor and indexed work area.
Emulation	An activity of imitating the behavior of real cell or line and display the activity on screen.
Ghost picking	A kind of dry run, when production uses recorded virtual items to pick, thus no real item to pick.
Offline Simulation	Simulation process when connected to the virtual robot.

This page is intentionally left blank

2 Getting started

How to start PickMaster Recipe Manager

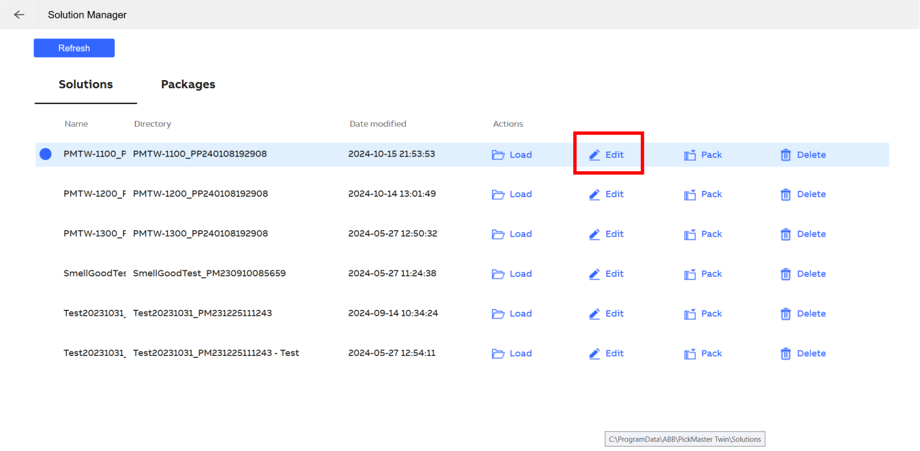
The PickMaster Recipe Manager can only be started through PickMaster Twin Operator.

There are two methods to start PickMaster Recipe Manager.

Method 1

1 Open PickMaster Twin Operator.

2 Go to **Solution Manager** page, and click **Edit** on the target solution.



Name	Directory	Date modified	Actions			
PMTW-1100_F	PMTW-1100_PP240108192908	2024-10-15 21:53:53	Load	Edit	Pack	Delete
PMTW-1200_F	PMTW-1200_PP240108192908	2024-10-14 13:01:49	Load	Edit	Pack	Delete
PMTW-1300_F	PMTW-1300_PP240108192908	2024-05-27 12:50:32	Load	Edit	Pack	Delete
SmellGoodTes	SmellGoodTest_PM230910085659	2024-05-27 11:24:38	Load	Edit	Pack	Delete
Test20231031_	Test20231031_PM231225111243	2024-09-14 10:34:24	Load	Edit	Pack	Delete
Test20231031_	Test20231031_PM231225111243 - Test	2024-05-27 12:54:11	Load	Edit	Pack	Delete

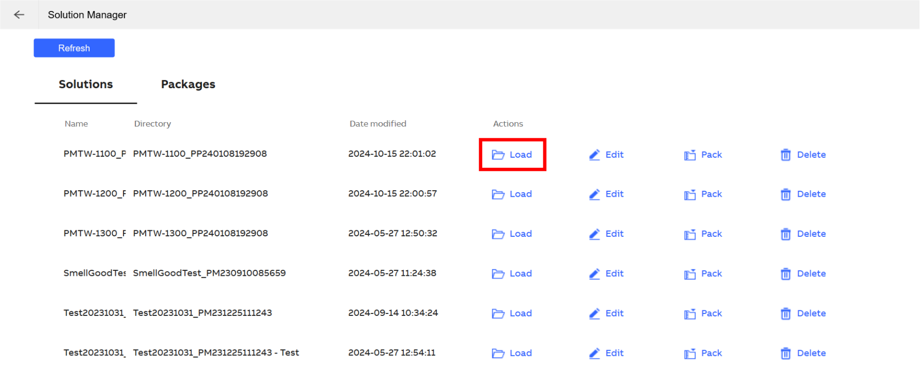
C:\ProgramData\ABB\PickMaster Twin\Solutions

xx2400001758

Method 2

1 Open PickMaster Twin Operator.

2 Go to **Solution Manager** page, and click **Load** on the target solution.

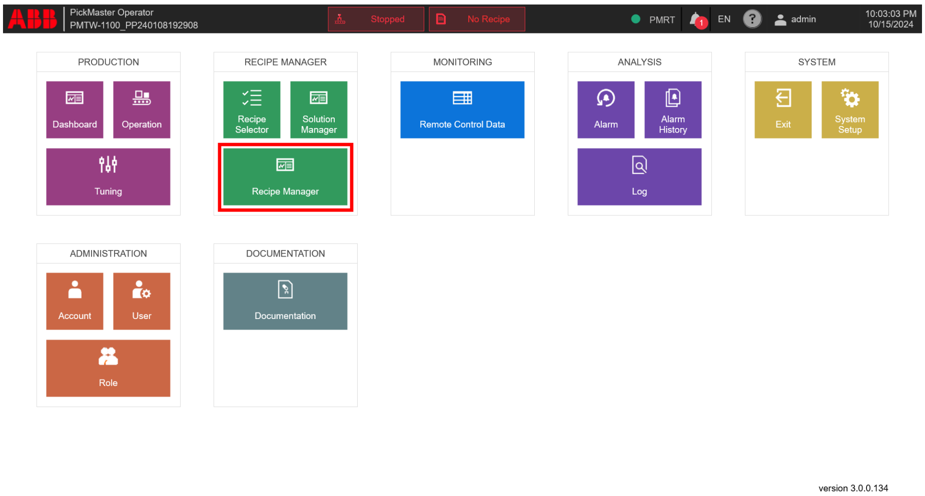


Name	Directory	Date modified	Actions			
PMTW-1100_F	PMTW-1100_PP240108192908	2024-10-15 22:01:02	Load	Edit	Pack	Delete
PMTW-1200_F	PMTW-1200_PP240108192908	2024-10-15 22:00:57	Load	Edit	Pack	Delete
PMTW-1300_F	PMTW-1300_PP240108192908	2024-05-27 12:50:32	Load	Edit	Pack	Delete
SmellGoodTes	SmellGoodTest_PM230910085659	2024-05-27 11:24:38	Load	Edit	Pack	Delete
Test20231031_	Test20231031_PM231225111243	2024-09-14 10:34:24	Load	Edit	Pack	Delete
Test20231031_	Test20231031_PM231225111243 - Test	2024-05-27 12:54:11	Load	Edit	Pack	Delete

xx2400001759

3 Back to **Home** page, and click **Recipe Manager**.

Continues on next page



xx2400001760

Then the PickMaster Recipe Manager will be opened with the desired solution.

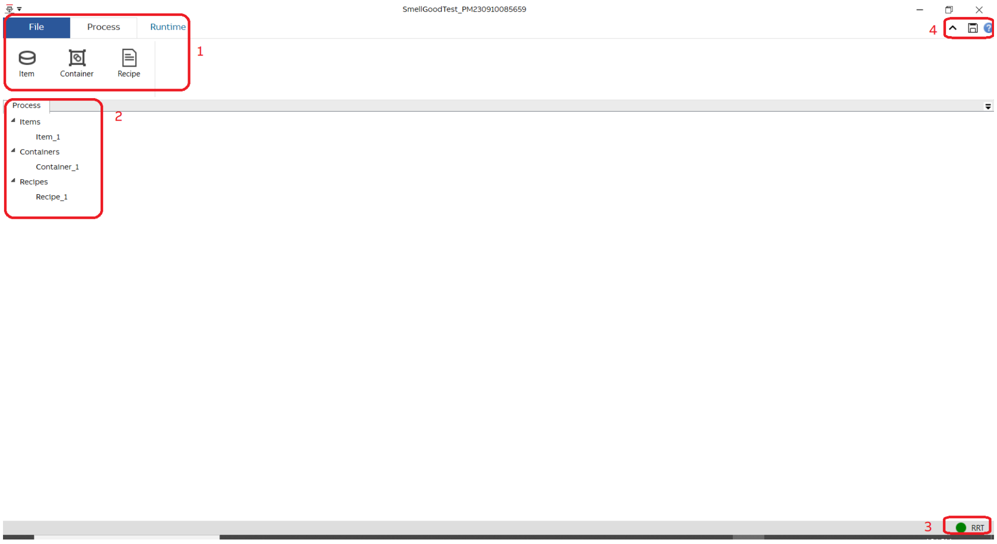
When a solution is opened with PickMaster Recipe Manager, this solution will be unloaded from the PickMaster Operator automatically.

3 Navigating PickMaster Recipe Manager

3.1 Main window

Overview

This chapter describes about the user interface of the PickMaster Recipe Manager. The following figure and table provides information regarding the major elements in the user interface.



xx2300002044

		Description
1	Ribbon tab	Contains the general functions for PickMaster Recipe Manager. When creating a new solution, the work flow is usually from left to right. For more details, see the section *Ribbon tab on page 20*.
2	Tree view browser	Organizes the programmable objects (for example, items, container, and recipes) of the picking application in a tree structure. For more details, see the section *Tree view browser on page 23*.
3	Status view	Shows the status of connecting with Runtime at present.
4	Additional operation view	Shows the save button and help button. **Help**: open the PickMaster Recipe Manager operating manual. xx2100000867

Tip

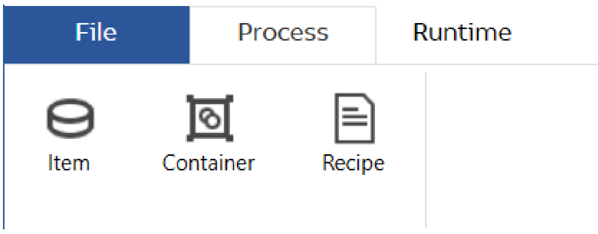
All windows can be distributed and floating freely.

3.2 Ribbon tab

Overview

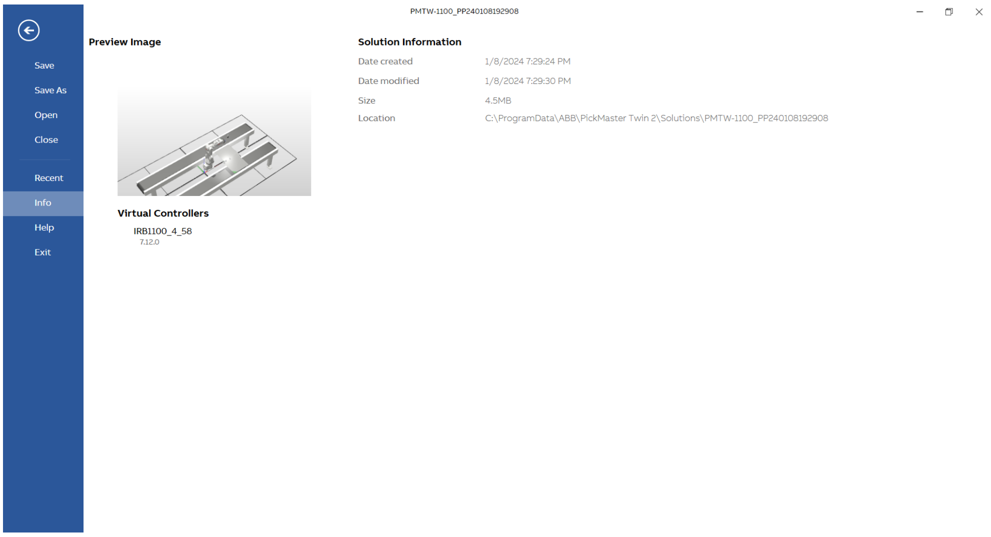
The PickMaster Recipe Manager ribbon contains elements arranged in various groups. The following figures and tables provide more information regarding the elements in the PickMaster Recipe Manager ribbon.

Following are the objects and configurations saved in the ribbon tab.



xx2300002045

File



PMTW-1100_PP240108192908

Preview Image

Virtual Controllers

IRB1100_4_58
7.12.0

Solution Information

Date created 1/8/2024 7:29:24 PM

Date modified 1/8/2024 7:29:30 PM

Size 4.5MB

Location C:\ProgramData\ABB\PickMaster Twin 2\Solutions\PMTW-1100_PP240108192908

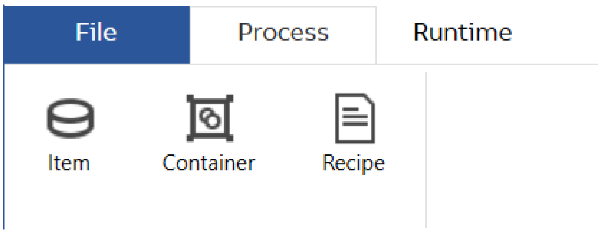
xx2300002046

Button	Description
xx2100000857	Go back to the main window.
Save	Save the changes for the solution at present. **Note** If the solution will be used in the PickMaster Operator, it must have been connected to a real controller with the same configuration on PickMaster PowerPac.

Continues on next page

Button		Description
Save as		Save your present solution as a new solution in desired location. **Note** If the solution will be used in the PickMaster Operator, it must have been connected to a real controller with the same configuration on PickMaster PowerPac.
Open		Open other solutions or any solutions saved in your local folder. **Tip** Only solutions or shared files which are created with PickMaster PowerPac 2.0 or later can be opened.
Close		Close your present solution.
Recent		Open the solutions which has been opened before.
Info		Show the basic information of the opened solution. **Tip** This page will only show up when a solution is opened.
Help	**About**	Shows the basic version information.
	Options	**Language**: Will follow PickMaster Operator language setting **Rapid Editor**: specify the editor to open Rapid. **Note** If the user changes the language in PickMaster Twin Operator during working with PickMaster Recipe Manager , the selected language will be valid after the PickMaster Recipe Manager restarted.
	Manual	Open the PickMaster Recipe Manager operating manual.
Exit		Close and exit the PickMaster Recipe Manager.

Process



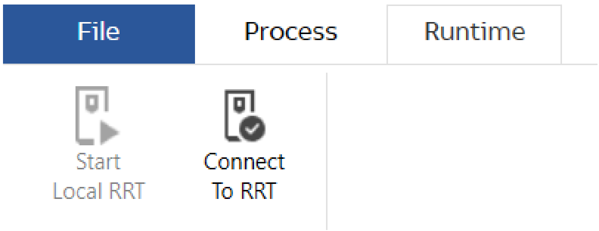
xx2300002045

Button	Description
Items	Add items. More details about creating an item is available in the section *Adding an item on page 27*.

Continues on next page

Button	Description
Container	Add containers. More details about creating an container is available in the section *Adding a container on page 34*.
Recipe	Create a recipe. More details about creating a recipe is available in the section *Adding a recipe on page 42*.

Runtime



xx2300002047

Button	Description
Start Local RRT	Start the **Runtime** on the computer. **Local RRT** means the Runtime installed with PickMaster Operator.
Connect to RRT	Connect to the real **Runtime**.

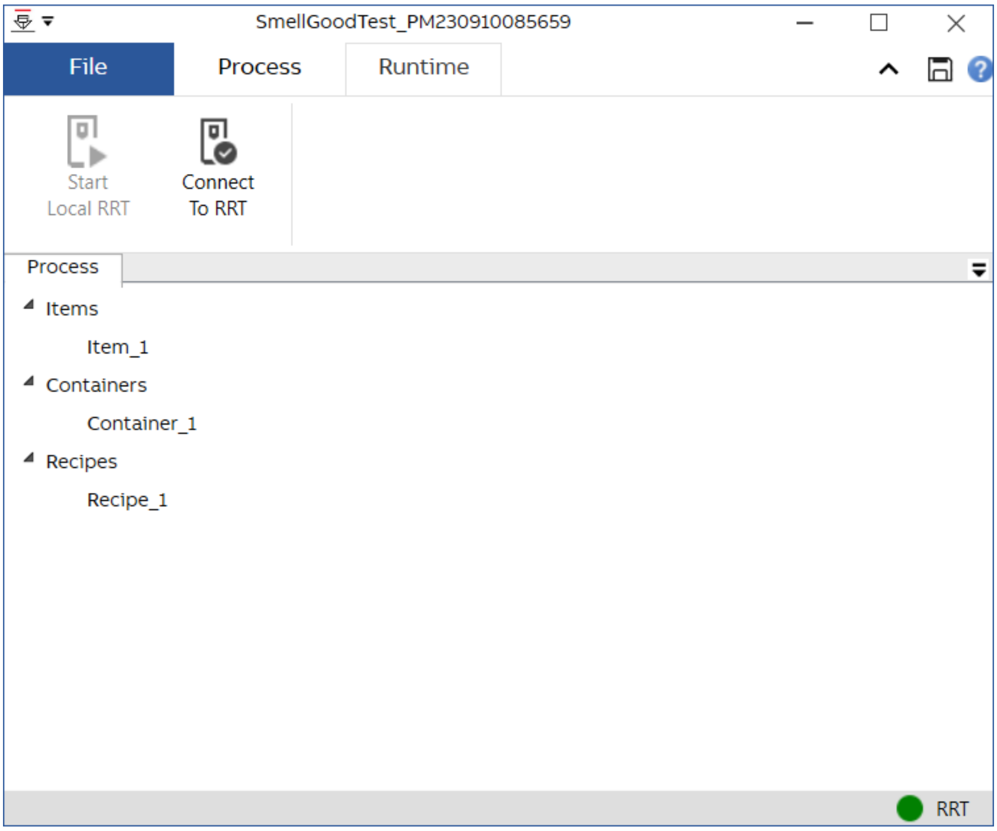
3.3 Tree view browser

3.3.1 Process

Overview

The **Process** tab displays the configuration file and the application hardware objects such as items, containers and recipes.

Following are the objects and configurations saved in the **Process** tab.



xx2300002048

- **Items**
- **Containers**
- **Recipes**

Items

Managing item

Right-click on an **item** in the tree view to manage the item.

	Description
Setting	Manage the settings of the selected item. When you select **Setting**, the **Item Setting** window is displayed. More details about managing a selected item is available in the section *Adding an item on page 27*.
Delete	Delete the selected item.

Continues on next page

	Description
Rename	Change the name of the selected item.
Copy	Create a copy of the selected item with all settings.

Containers

Managing container

Right-click on a **Container** in the tree view to manage the container.

	Description
Setting	Manage the settings of the selected container. When you select **Setting**, the **Container Setting** window is displayed. More details about managing a selected container is available in the section *Adding a container on page 34*.
Delete	Delete the selected container.
Rename	Change the name of the selected container.
Copy	Create a copy of the selected container with all settings.

Recipes

Managing recipe

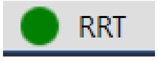
Right-click on a **Recipe** in the tree view to manage the recipe.

	Description
Setting	Manage the settings of the selected recipe. When you select **Setting**, the **Recipe Setting** window is displayed. More details about managing a selected recipe is available in the section *Adding a recipe on page 42*.
Delete	Delete the selected recipe.
Rename	Change the name of the selected recipe.
Copy	Create a copy file of the selected recipe with all settings.

3.4 Status view

Status

When the system starts, the status of the Runtime will show up on the bottom right corner as the illustration.



xx2300002049

	Description	Note
RRT	**Red**: The connection to the real Runtime fails. **Green**:The connection to the real Runtime successes.	

This page is intentionally left blank

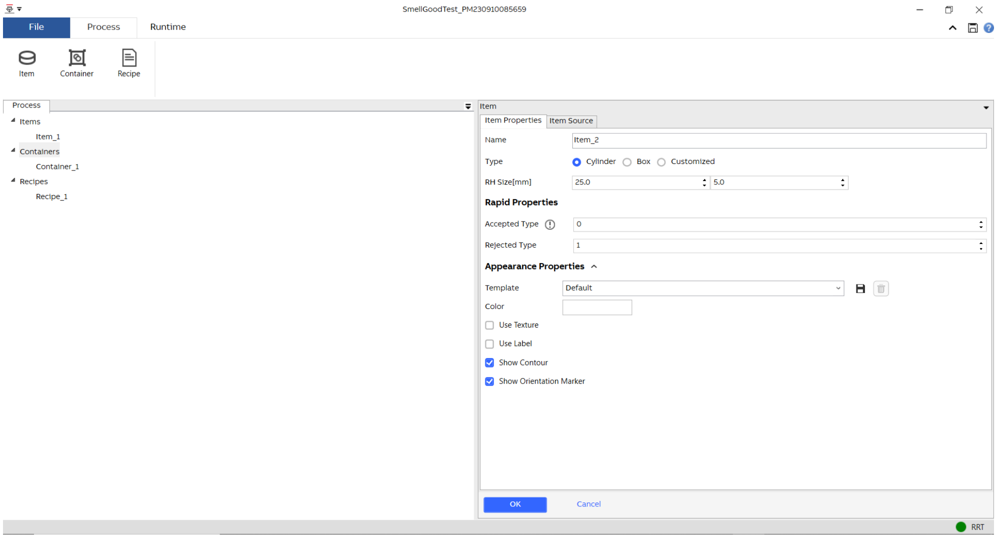
4 Workflow for PickMaster Recipe Manager

4.1 How to add or modify an item

Adding an item

Click **Item** on the ribbon to add an item in the solution.

The following table provides details about the **Item** setting dialog box.



Name: Item_2

Type: ● Cylinder ○ Box ○ Customized

RH Size[mm]: 25.0 | 5.0

Rapid Properties

Accepted Type: 0

Rejected Type: 1

Appearance Properties

Template: Default

Color

☐ Use Texture

☐ Use Label

☑ Show Contour

☑ Show Orientation Marker

xx2300002050

Item Properties tab

Item Properties

	Description
Name	Change the name.
Type	Change the shape of the item. - **Cylinder** - **Box** - **Customized**: import predefined models. **Tip** Only `*.rslib` and `*.rsgfx` can be imported into PickMaster Recipe Manager. These two types files are exported from Robotstudio.
Size(x,y,z)[mm]	Configure the size of the item.

Rapid properties

	Description
Accepted Type	Define the values for accepted item types. The values for the accepted item type are sent to the RAPID program and are supplied with the item targets. For more details see, *GetItmTgt - Get the next item target* in PickMaster® Twin - PowerPac Application manual.

Continues on next page

	Description
Rejected Type	Define the values for rejected item types. The values for the rejected item type are sent to the RAPID program and are supplied with the item targets. For more details see, *GetItmTgt - Get the next item target* in PickMaster® Twin - PowerPac Application manual.

Note

If the **Accepted Type** or **Rejected Type** of different items in one solution set as the same value, the **Picking Status** will be influenced.

Appearance Properties

	Description
Template	**Default Settings** tab: choose one of the preset templates. **Default Name** text box: enter the name for a new template. **Save** icon: save your new template. **Delete** icon: delete your templates. **Tip** If you enter a new template name in the template text box, a new template will be created instead of being renamed. **Note** If you directly modify the appearance of the default template instead of creating a new template, this will modify the default value of the default template. And all items created with default template will be modified too.
Color	Change the color of the new item.
Use Texture	Use a texture image file for the item.
Label Location	Set the location of the label on the item.
Label Picture	Select an image file for the label picture.
Show Contour	Choose to show the contour or not.
Show Orientation Marker	Choose to show the orientation maker or not.
Browse	Select and import a **Customized** model.
Offset [mm]	Set the offset value for the imported **Customized** models.
Orientation [deg]	Set orientation for the imported **Customized** models.

Item Source tab

Note

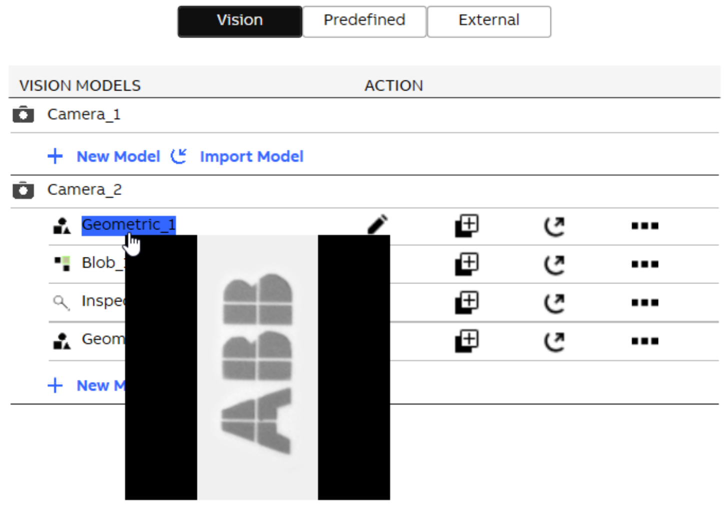
If the user changes the source type of an item, the user need to redo the selection in the related recipe setting accordingly.

Continues on next page

	Description
Vision	If the source type is set to **Vision**, a camera and vision models are used to find the object positions. The vision models are described in section *Adding vision model on page 55*. For more information regarding Vision Models see the following section.
Predefined	If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used.
External	If the source type is set to **External**, an external sensor in the solution together with external position generators are used to define item positions. The external sensor can only be created and configured in PickMaster PowerPac. For more information, see *Configuring external sensor* in PickMaster® Twin - PowerPac Application manual.

Continues on next page

Vision

	Description
New Model	Add a new vision model. • **Geometric**: Add a geometric vision model. A geometric sub inspection model is configured in the same way as a *PatMax* model. See *Configuring a geometric model on page 58*. In addition, the relative positions of the found items and the corresponding alignment hit must be trained. • **Blob**: Add a blob vision model. A blob sub inspection model is configured in the same way as a blob model. See *Configuring blob models on page 67*. In addition, the number of required hits must be configured. • **Inspection**: Add an inspection vision model. When hovering over the vision model name for one second, the trained model will be displayed as a preview image. **Note** Only geometric model or inspection model with geometric alignment model can be previewed. **Note** All the vision models created with PickMaster Powerpac 2.3.1 or lower version cannot be previewed directly. Open the edit tab and click **OK** button to generate the preview image when processing the vision models created with PickMaster Powerpac 2.3.1 or lower version.  Vision Predefined External VISION MODELS ACTION Camera_1 New Model Import Model Camera_2 Geometric_1 Blob_ Inspe Geom New M xx2400000635
Import Model	Import an existed vision model.
External	Add an external sensor. An external sensor is a software component that gives external partners full control of how item positions are generated. An external sensor can use any type of item detection such as barcode readers, cameras, or a combination of photo sensors to generate item positions.
Edit	Edit the selected vision model.

Continues on next page

	Description
Copy	- **Copy**: Copy the selected vision model to a same type model. - **Copy as an inspection model**: Copy the selected vision model and save as an inspection model with the selected vision model as the alignment model. **Note** For a geometric model, only geometric model with item height setting can be copied as inspection model. Vision height and external height can only be used in geometric model. They are not available for inspection model with geometric alignment model. Then a geometric model with vision height or external height setting cannot be copied to an inspection model. For more information about item height, vision height and external height, see *Configuring height settings on page 111*.
Export	Export the selected vision model.
More	- **Delete**: Delete the selected vision model. - **Rename**: Rename the selected vision model.

Predefined

	Description
Position(X,Y,Z)[mm]	Set the position for the predefined model.
Angle Z[deg]	Set the angle on Z axis of the predefined model.

External

Tip

The **External** configuration for items/containers can only be implemented when real Runtime is connected.

	Description
New position generator	Add an external position generator. When users have not created the position generator for this sensor before, they have to click the **new position generator** button first. Then the python interface of `def configurePosGen(self, posGenId)` will be automatically called, which is the same as the next operation “Configure”. The prerequisite of this operation is that the corresponding external sensor has already been configured according to section 3.2, otherwise there will be a message box showing “The current sensor is not configured. Please configure the sensor before creating the position generator.” The external sensor can only be created and configured in PickMaster PowerPac. For more information, see *Configuring external sensor* in PickMaster® Twin - PowerPac Application manual.
SYNC TIME[MS]	The time of RT received strobe signal is calculated by the current system time (StrobeTime) minus the time of data process(iTimeSinceStrobe). But the time of from controller trigger strobe signal to RT received strobe signal cannot be calculated. So the value of Synchronization time is used to compensate for this value. This value will be set by users to compensate the time spent for signal transmission on hardware and invoking function. For different external sensor, this value may be set differently.

Continues on next page

	Description
Configure	Once the position generator is created and configured, users could click the button of **Configure** to do configuration again. This operation refers to the Python interface of `def configurePosGen(self, posGenId)`. Users should self-define the position generator configuration behavior in this interface in their own Python class. Although users could only create one position generator in PMPP UI, users could implement more position generation methods in this interface, so that positions could be generated based on one or more methods. The same as sensor configuration, the position generator configuration information should be serialized into a string, so that PMPP solution could get and save this string. This button could be clicked as long as its button state is enabled. If the current row is in disabled state, the corresponding position generator could not be configured until it enters configuration – enabled state.
Delete	Delete the selected position generator.
Save	In the save – enabled state, users could click “Save” button to get the configuration string from the Python program and update in PMPP. This button refers to the Python interface “def savePosGen(self, posGenId)” which is provided by PMTW developer in ExternalSensorInterface.py file and users should not modify the interface content. The content only contains returning the configuration string, so users should make sure that all configured information are included in this string in the “configurePosGen” interface. After “Save” button is clicked, all rows will enter configuration - enabled state.
OK	The “OK” button is for the item/container view. When this button is clicked, all data will be saved, and the item/container view will be closed. If one external sensor position generator is in save – enabled state, the “savePosGen” Python interface will firstly be called before the view is closed.
`Cancel`	The “Cancel” button is for the item/container view. When this button is clicked, all modified data will be abandoned, and the item/container view will be closed.

Procedure

On the PickMaster Recipe Manager ribbon-tab, click **Process**.

Use this procedure to add an item:

1. On the ribbon-tab, click **Item**.

   The **Item** window opens.
2. In the **RH Size** part, define the item’s size.

   The height of the item defines the pick height and is always added to items found by a vision model or a position defined by a predefined position source.
3. If needed, define levels for accepted or rejected item types.

   When inspection is used, a found item will be marked as either accepted or rejected. The values for accepted and rejected item type in the **Item Configuration** dialog are sent to the RAPID program and are processed there. See *Configuring inspection models on page 73*.
4. Click **OK**.

Continues on next page

Modifying an item

Managing item

Right-click on an **item** in the tree view to manage the item.

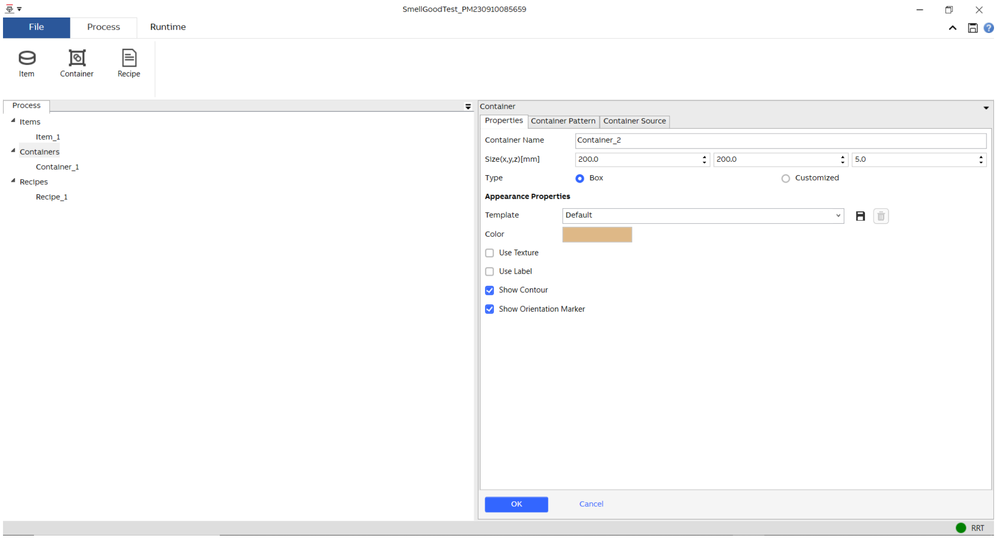
	Description
Setting	Manage the settings of the selected item. When you select **Setting**, the **Item Setting** window is displayed. More details about managing a selected item is available in the section *Adding an item on page 27*.
Delete	Delete the selected item.
Rename	Change the name of the selected item.
Copy	Create a copy of the selected item with all settings.

4.2 How to add or modify a container

Adding a container

Click **Container** on the ribbon to add a container in the solution.

The following table provides details about the **Container** setting dialog box.



Container Name: Container_2

Size(x,y,z)[mm]: 200.0 200.0 5.0

Type: ● Box ○ Customized

Appearance Properties

Template: Default

Color

☐ Use Texture

☐ Use Label

☑ Show Contour

☑ Show Orientation Marker

xx2300002051

Container Properties tab

Container Properties

	Description
Container Name	Change the name.
LWH Size (mm)	Configure the size of the container.
Type	Define the type of the container. • **Box** • **Customized**: import predefined models. **Tip** Only `*.rslib` and `*.rsgfx` can be imported into PickMaster Recipe Manager. These two types files are exported from Robotstudio.

Continues on next page

Appearance Properties

	Description
Template	**Default Settings** tab: choose one of the preset templates. **Default Name** text box: enter the name for a new template. **Save** icon: save your new template. **Delete** icon: delete your templates. **Tip** If you enter a new template name in the template text box, a new template will be created instead of being renamed. **Note** If you directly modify the appearance of the default template instead of creating a new template, this will modify the default value of the default template. And all containers created with default template will be modified too.
Color	Change the color of the container.
Use Texture	Use a texture image file for the container.
Label Location	Set the location of the label on the container.
Label Picture	Select an image file for the label picture.
Show Contour	Choose to show the contour or not.
Show Orientation Marker	Choose to show the orientation maker or not.
Browse	Select and import a **Customized** model.
Offset [mm]	Set the offset value for the imported **Customized** models.
Orientation [deg]	Set orientation for the imported **Customized** models.

Container Pattern tab

A pattern defines a collection of positions. For example, a box with predefined locations for certain objects. You can change the order, delete, or rearrange the selected layers using the available options. You can adjust the vertical position of each layer by modifying the **Offset** (mm). You can also manage the sorting method. The **Sorting Method** section defines the order in which the items in the container pattern shall be handled by the robots.

	Description
Add Layer	Add a new layer. For more information regarding Add Layer see the following section.
Edit Layer	Edit the selected layer.
Copy	Copy the selected layer.
Delete Layer	Delete the selected layer.
Up	Move the selected layer to a upper level.
Down	Move the selected layer to a lower level.
Delete All	Delete all the existing layers.

Continues on next page

	Description
Total Weight	Shows the total weight of all the items.
Total Height	Shows the total height of all the items.
Total Count	Shows the total count of all the items.

Add Layer

	Description
Available Items	Select one available item that has been created. **Add** icon: add the selected item onto the layer. **Delete** icon: delete the selected items. **Select All** icon: select all the items in the layer.
Align Style	Define the align style when you have more than one item in the layer. **Left Align** icon: align all the items in this layer from the left. **Center Align** icon: align all the items in this layer from the center. **Right Align** icon: align all the items in this layer from the right. **Top Align** icon: align all the items in this layer at from top. **Middle Align** icon: align all the items in this layer from the middle. **Bottom Align** icon: align all the items in this layer from the bottom.
Distribute Style	Define the distribution style when you have more than one item in the layer. **Horizontally** icon: distribute all the items in the horizontal direction. **Vertically** icon: distribute all the items in the vertical direction.
Else Functions	**Rotate** icon: rotate the selected items.
Sorting Method	Configure the signals. Use the **Customized Settings** options to manage the signals. **None** options: The items in the layer shall be accessed in the same order as they are defined in the layout for each layer, but if the next item cannot be reached the next one after that is used. **X Direction** options: The items shall be accessed in the X direction for each layer, that is, in the order they travel along a conveyor. **Strict** options: The items shall be used in the same order as they are defined in the layout for each layer. If a robot cannot access the next item position in a layer, that robot does not use any more item positions in the container pattern.
Order	Define the order of the layer.
Position X Y Z [mm]	Define the position of the item in the layer.
Angle X Y Z [deg]	Define the angle of the item in the layer.
Show Item Name	Shows the name of the items.
Show Item Order	Shows the added order of the items.

Container Source tab

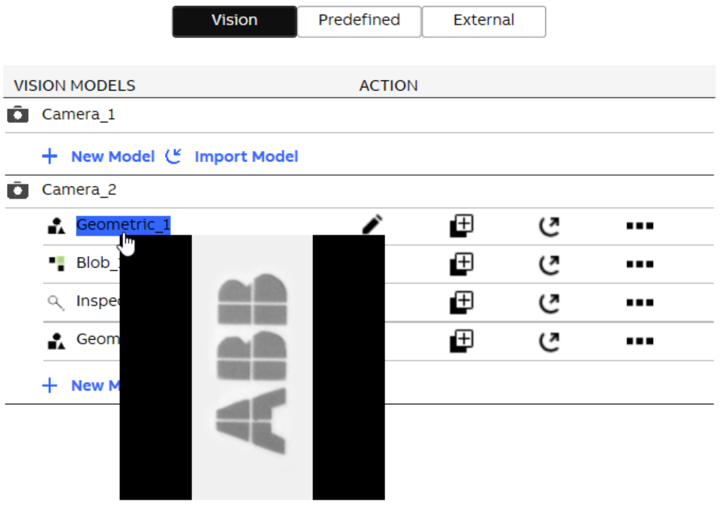
	Description
Vision	If the source type is set to **Vision**, a camera and vision models are used to find the object positions. The vision models are described in section *Adding vision model on page 55*. For more information regarding Vision Models see the following section.

Continues on next page

	Description
Predefined	If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used.
External	If the source type is set to **External**, an external sensor in the solution together with external position generators are used to define container positions. The external sensor can only be created and configured in PickMaster PowerPac. For more information, see *Configuring external sensor* in PickMaster® Twin - PowerPac Application manual.

Continues on next page

Vision

	Description
New Model	Add a new vision model. • **Geometric**: Add a geometric vision model. A geometric sub inspection model is configured in the same way as a *PatMax* model. See *Configuring a geometric model on page 58*. In addition, the relative positions of the found items and the corresponding alignment hit must be trained. • **Blob**: Add a blob vision model. A blob sub inspection model is configured in the same way as a blob model. See *Configuring blob models on page 67*. In addition, the number of required hits must be configured. • **Inspection**: Add an inspection vision model. When hovering over the vision model name for one second, the trained model will be displayed as a preview image. **Note** Only geometric model or inspection model with geometric alignment model can be previewed. **Note** All the vision models created with PickMaster Powerpac 2.3.1 or lower version cannot be previewed directly. Open the edit tab and click **OK** button to generate the preview image when processing the vision models created with PickMaster Powerpac 2.3.1 or lower version.  Vision Predefined External VISION MODELS ACTION Camera_1 New Model Import Model Camera_2 Geometric_1 Blob_ Inspe Geom New M xx2400000635
Import Model	Import an existed vision model.
External	Add an external sensor. An external sensor is a software component that gives external partners full control of how item positions are generated. An external sensor can use any type of item detection such as barcode readers, cameras, or a combination of photo sensors to generate item positions.
Edit	Edit the selected vision model.

Continues on next page

	Description
Copy	- **Copy**: Copy the selected vision model to a same type model. - **Copy as an inspection model**: Copy the selected vision model and save as an inspection model with the selected vision model as the alignment model. **Note** For a geometric model, only geometric model with item height setting can be copied as inspection model. Vision height and external height can only be used in geometric model. They are not available for inspection model with geometric alignment model. Then a geometric model with vision height or external height setting cannot be copied to an inspection model. For more information about item height, vision height and external height, see *Configuring height settings on page 111*.
Export	Export the selected vision model.
More	- **Delete**: Delete the selected vision model. - **Rename**: Rename the selected vision model.

Predefined

	Description
Position(X,Y,Z)[mm]	Set the position for the predefined model.
Angle Z[deg]	Set the angle on Z axis of the predefined model.

External

Tip

The **External** configuration for items/containers can only be implemented when real Runtime is connected.

	Description
New position generator	Add an external position generator. When users have not created the position generator for this sensor before, they have to click the **new position generator** button first. Then the python interface of `def configurePosGen(self, posGenId)` will be automatically called, which is the same as the next operation “Configure”. The prerequisite of this operation is that the corresponding external sensor has already been configured according to section 3.2, otherwise there will be a message box showing “The current sensor is not configured. Please configure the sensor before creating the position generator.” The external sensor can only be created and configured in PickMaster PowerPac. For more information, see *Configuring external sensor* in PickMaster® Twin - PowerPac Application manual.
SYNC TIME[MS]	The time of RT received strobe signal is calculated by the current system time (StrobeTime) minus the time of data process(iTimeSinceStrobe). But the time of from controller trigger strobe signal to RT received strobe signal cannot be calculated. So the value of Synchronization time is used to compensate for this value. This value will be set by users to compensate the time spent for signal transmission on hardware and invoking function. For different external sensor, this value may be set differently.

Continues on next page

	Description
Configure	Once the position generator is created and configured, users could click the button of **Configure** to do configuration again. This operation refers to the Python interface of `def configurePosGen(self, posGenId)`. Users should self-define the position generator configuration behavior in this interface in their own Python class. Although users could only create one position generator in PMPP UI, users could implement more position generation methods in this interface, so that positions could be generated based on one or more methods. The same as sensor configuration, the position generator configuration information should be serialized into a string, so that PMPP solution could get and save this string. This button could be clicked as long as its button state is enabled. If the current row is in disabled state, the corresponding position generator could not be configured until it enters configuration – enabled state.
Delete	Delete the selected position generator.
Save	In the save – enabled state, users could click “Save” button to get the configuration string from the Python program and update in PMPP. This button refers to the Python interface “def savePosGen(self, posGenId)” which is provided by PMTW developer in ExternalSensorInterface.py file and users should not modify the interface content. The content only contains returning the configuration string, so users should make sure that all configured information are included in this string in the “configurePosGen” interface. After “Save” button is clicked, all rows will enter configuration - enabled state.
OK	The “OK” button is for the item/container view. When this button is clicked, all data will be saved, and the item/container view will be closed. If one external sensor position generator is in save – enabled state, the “savePosGen” Python interface will firstly be called before the view is closed.
`Cancel`	The “Cancel” button is for the item/container view. When this button is clicked, all modified data will be abandoned, and the item/container view will be closed.

Procedure

On the PickMaster Recipe Manager ribbon-tab, click **Process.**

Use this procedure to add a container :

1 On the ribbon-tab, click **Container**.

The **Container** window opens.

2 Define the container with your requirements in the **Container Properties** tab.

3 Define the container pattern with your requirements in the **Container Pattern** tab.

4 In the **Container Pattern** tab, click **New Layer** to define a layer in the container.

5 If need, adjust the layout of the items on the layer.

A Select all items on the layer.

B Click 'Ctrl' and select the base item at the same time.

C Click **Left** to align all items on the left edge according to the base item.

Click **Right** to align all items on the right edge according to the base item.

Continues on next page

Click **Center** to align all items on the centre line vertically according to the base item.

Click **Middle** to align all items on the centre line horizontally according to the base item.

Click **Top** to align all items on the top edge according to the base item.

Click **Bottom** to align all items on the bottom edge according to the base item.

D Click **Horizontally** to set all items tangent in horizontal direction.

Click **Vertically** to set all items tangent in vertical direction.

6 Click **Save**.

The layer layout is saved.

7 Click **OK**.

The container is saved and the window is closed.

Modifying a container

Managing container

Right-click on a **Container** in the tree view to manage the container.

	Description
Setting	Manage the settings of the selected container. When you select **Setting**, the **Container Setting** window is displayed. More details about managing a selected container is available in the section *Adding a container on page 34*.
Delete	Delete the selected container.
Rename	Change the name of the selected container.
Copy	Create a copy of the selected container with all settings.

4.3 How to add or modify a recipe

4.3.1 Adding a recipe

Overview

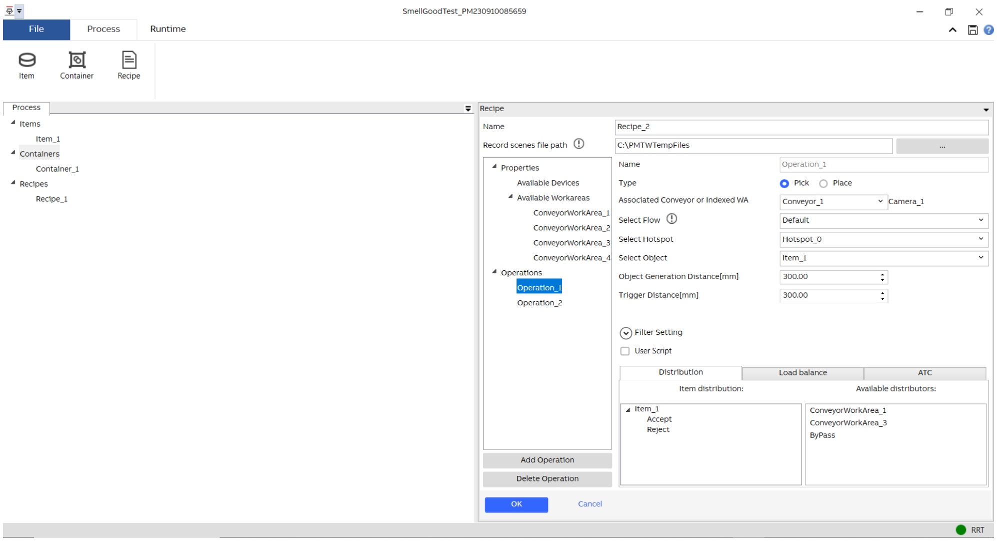
This section describes how to add a recipe.

In one solution, several recipes can be created. All elements (Robots, sensor and so on) in this solution can be added to any recipes with no limits.

Adding a recipe

Click **Recipe** on the ribbon to add a recipe in the solution.

The following table provides details about the **Recipe** setting dialog box.



Name: Recipe_2
Record scenes file path: C:\PMTWTempFiles
Name: Operation_1
Type: ● Pick ○ Place
Associated Conveyor or Indexed WA: Conveyor_1 Camera_1
Select Flow: Default
Select Hotspot: Hotspot_0
Select Object: Item_1
Object Generation Distance[mm]: 300.00
Trigger Distance[mm]: 300.00
Filter Setting
□ User Script

xx2300002052

Properties

	Description
Available Devices	Define the available devices, including robots and conveyors. All robots and conveyors in the same solution will be listed in every recipe, but they can have different attribute settings in different recipes. For example, the speed of the same robot can be different in different recipes. For more information regarding Available Device see the following section.
Available Workareas	Define the available work areas. All work areas in the same solution will be listed in every recipe, but they can have different attribute settings in different recipes. For more information regarding Available Work Areas see the following section.

Continues on next page

Available Devices

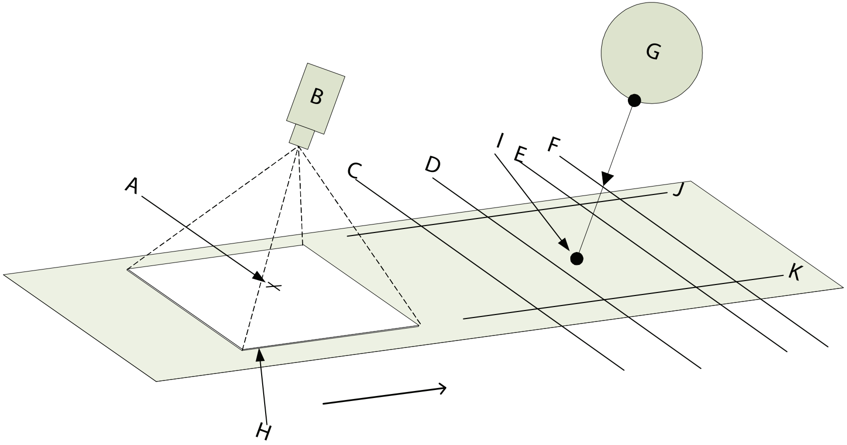
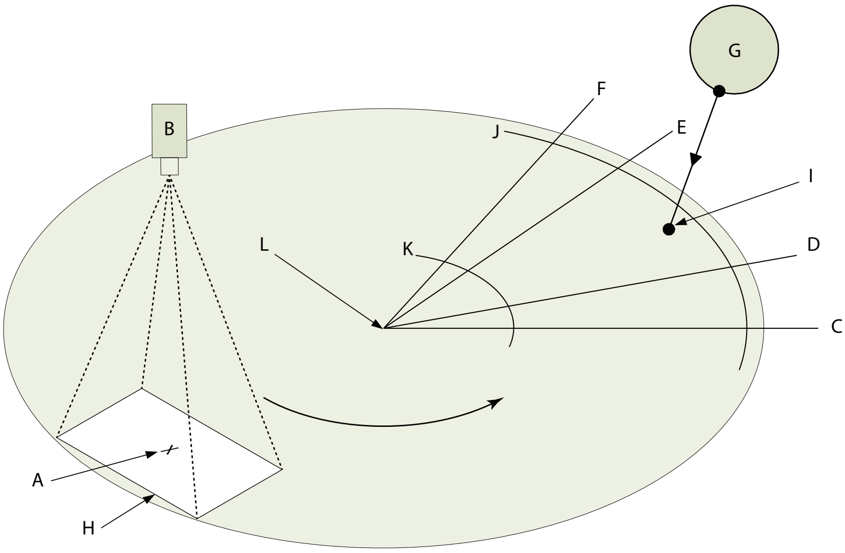
	Description
Robot Setting	**Note** If there are more than one robot in this system, all the robot will be listed here with their defined name. **Speed**: change the speed of the robot. **Rapid**: import/export/edit the Rapid program of the robot. **Rapid Editor**: specify the editor to open Rapid. **Note** The default RAPID module is created for IRB 360. Alternative RAPID template modules for different robot type categories and for double picking can be imported from the installation folder: `C:\Program Files (x86)\ABB\PickMaster Twin 3\Samples\RAPIDs`.
Conveyor Setting	**Speed**: change the speed of the conveyor. **Acceleration**: change the acceleration of the conveyor. **Deceleration**: change the deceleration of the conveyor.
Cameras Parameter Setting	**Sync All Cameras**: synchronize the parameter of all cameras from **Camera Configuration** to recipe camera parameter setting. **Sync**: synchronize the parameter of the camera from **Camera Configuration** to recipe camera parameter setting. **Reset**: reset the parameters to the last saved data. **Exposure**: the exposure of the camera. **Brightness**: the brightness of the camera. **Contrast**: the contrast of the camera. **Live/Stop**: show/stop the live video image dialog with current setting. **Apply**: apply the parameter and take effect in the live video image dialog.

Continues on next page

Available Work Areas

	Description	
Pick Setting	**Pick/place elevation**	The distance, in negative z-direction relative to the tool, from where the robot approaches the item target.
	Pick/place time[s]	The time that the robot is in the pick/place position. If the conveyor is moving during the pick/place time, the robot will track along the conveyor to keep the relative position on the moving conveyor.
	Vacuum Activation[s]	The time in seconds before the middle of the corner path of the approaching position, when the vacuum I/O should be set. If a negative value is entered, the vacuum I/O will be set the time after the middle of the corner path. This value is only valid for work areas of type **Pick**. **Note** Vacuum activation does not affect the picking of items in simulation. Items are attached to the picking tool using `SimAttach` events, for example, in the Pick Routine.
	Vacuum Reversion[s]	The time in seconds before the half place time in the place position, when the blow I/O should be set. If a negative value is entered, the blow I/O will be set the time after the half place time in the place position. This value is only valid for work areas of type **Place**. **Note** Vacuum reversion does not affect the placing of items in simulation. Items are detached from the picking tool using `SimDetach` events, for example, in the Place Routine.
	Vacuum Off[s]	The time in seconds after the half place time in the place position, when the blow I/O should be reset. If a negative value is entered, the blow I/O will be reset the time before the half place time in the place position. This value is only valid for work areas of type **Place**. **Note** Vacuum Off does not affect the placing of items in simulation. Items are detached from the picking tool using **SimDetach** events, for example, in the Place Routine.
	Load Time[s]	The generation interval time of the objects in the indexed work area. This value is only valid for indexed work areas.

Continues on next page

	Description
Area Setting	After you define a start entry in a work area which may called Start X , you can define a same start entry which may called Start Y at the vertical direction of the Start X.  xx1800001747  xx2400000631

Continues on next page

	Description

A	Camera and Baseframe origin for linear conveyor Camera origin for circular conveyor
B	Camera
C	Enter
D	Start
E	Stop
F	Exit
G	Robot
H	Image frame
I	Center of Robot
J	Y Max/Radius Max
K	Y Min/Radius Min
L	Baseframe origin for circular conveyor

Note

The reference origin for **Enter**, **Exit**, **Start**, and **Stop** is **I** (Center of Robot). The reference base for **Y Max** and **Y Min** is the conveyor base frame.

Enter[mm][i]**/[degree]**[ii]	**Enter** is the limit from where the robot starts to execute item targets on the work area (Start X). The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor. Make sure that the enter limit can be reached by the robot.
Start[mm][i]**/[degree]**[ii]	**Start** is when the next item to execute on the conveyor is above this limit, the conveyor is started. The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor.
Stop[mm][i]**/[degree]**[ii]	**Stop** is when an item on the conveyor reaches this limit, the conveyor is stopped. The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor.
Exit[mm][i]**/[degree]**[ii]	**Exit** is the limit from where the robot considers an item target as lost on the work area (Start X). The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor. When the tracked item passes beyond this limit it will be dropped. This limit must be chosen well within the maximum reach of the robot. The robot must be able to reach this position from an arbitrary position in the robot's working area before the position is out of reach.
Y Max[mm][i]**/Radius Max[mm]**[ii]	**Y Max[mm]/Radius Max[mm]** is the limit from where robot considers an item target as lost on the work area in End Y.The distance is calculated in millimeter from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving vertical direction of the conveyor. Make sure that the **Y Max/Radius Max** can be reached by the robot. If the y coordinate value of the item's position is greater than the **Y Max/Radius Max** , the robot will not grab the item. So when the tracked item passes beyond this limit it will be dropped. This limit must be chosen well within the maximum reach of the robot.

Continues on next page

<table>
<tr><th></th><th colspan="2">Description</th></tr>
<tr><td></td><td><b>Y Min[mm]</b>[i]/ <b>Radius Min[mm]</b>[ii]</td><td><b>Y Min[mm]</b>/<b>Radius Min[mm]</b> is the limit from where robot starts to execute item targets on the work area in Start Y. The distance is calculated in millimeter from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving vertical direction of the conveyor.</td></tr>
<tr><td></td><td><b>Use Start/Stop</b></td><td>Select the this checkbox if the work area should supervise the start and stop limits.
<b>Note</b>
<b>Start</b> and <b>Stop</b> values should be within boundaries of <b>Enter</b> and <b>Exit</b> limits. The value of <b>Enter</b> MUST be smaller than the value of <b>Start</b>. The value of <b>Stop</b> MUST be smaller than the value of <b>Exit</b>.
Otherwise there will be some errors during simulation.
<b>Note</b>
When <b>Use Start/Stop</b> checkbox is selected, the distance between <b>Stop</b> and <b>Exit</b> should be larger than the size (x direction) of the container .
This is handled by the <i>Conveyor start/stop</i> signal, see <i>Adding the conveyor work area</i> in PickMaster® Twin - PowerPac Application manual.</td></tr>
<tr><td></td><td><b>Start with production</b></td><td>Select the this checkbox if the work area should work with the conveyor when the production is started, and stopped when the production is stopped.</td></tr>
<tr><td></td><td><b>Use Y Max/Y Min</b>[i]/<b>Use Radius Max/Radius Min</b>[ii]</td><td>Select the this checkbox if the work area should supervise the upper and lower limits.</td></tr>
<tr><td><b>Record Setting</b></td><td colspan="2">Record the position of the items and containers in simulation and production.
<b>Note</b>
When <b>Record scenes</b> is selected and saved for any work area, the following message will pop up.
<code>Scenes recording is activated for: {0}</code>
After this, the recording will be activated automatically when the simulation or production is started.
The scenes for all work area with the same operation will be saved in one <code>.xml</code> file.</td></tr>
</table>

[i] Only available when the conveyor is linear conveyor.

[ii] Only available when the conveyor is circular conveyor.

Operation

The operation contains pick operation and place operation.

	Description
Main Setting	Define some basic settings for the operation, such as operation name, flow, source type. For more information regarding Main Setting see the following section.
Filter Setting	Define the filter setting for the operation. For more information regarding Filter Setting see the following section.

Continues on next page

	Description
User Script	Select to define the **User Script** function for the operation. For more information regarding User Script see the following section.
Distribution Setting	Define the distribution setting for the operation. For more information regarding Distribution Setting see the following section.

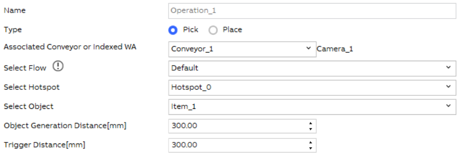
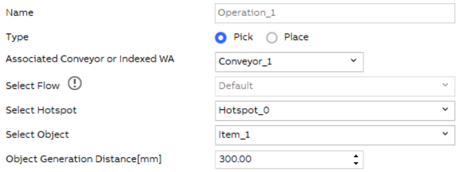
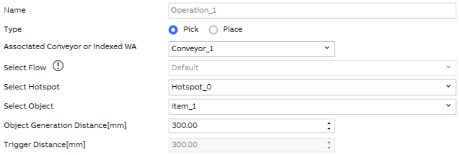
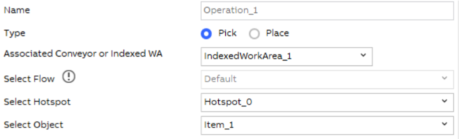
Main Setting

	Description
Operation Name	Rename the operation.
Operation Type	Set the type of the operation.
Associated Conveyor or Indexed WA	Select the associated conveyor or indexed WA.
Select Flow	Select the flow you defined. For more detail on how to add a flow, see *Adding Flow* in PickMaster® Twin - PowerPac Application manual. **Tip** If an external sensor is used on the conveyor, Flow function will be disabled.
Select Hotspot	Select the hotspot you defined.
Select Object	Select the available items or containers you defined.
Object Generation Distance[mm]/[degree]	Define the object generated distance value. **Tip** If an indexed work area is used, **Object Generation Distance[mm]/[degree]** is not available. For more information, see the following table.
Trigger Distance[mm]/[degree]	Define the trigger distance value when **Trigger Setting** is set as **Distance**. **Note** When **Source Type** is set as **Predefined** and **Trigger Setting** is set as **Distance**, the trigger distance value comes from the **Object Generation Distance[mm]/[degree]** value. For more information, see the following table.

Continues on next page

Different conditions for using **Object Generation** and **Trigger Distance**

As the **Object Generation Distance[mm]**/**[degree]** and **Trigger Distance[mm]**/**[degree]** are valid in different conditions, we list all conditions with their different options as below:

	Source Type	Trigger Setting	Object Generation Distance[mm]/[degree]	Trigger Distance[mm]/[degree]	Main Setting view
Conveyor	**Vision/External Sensor**	**Distance**	Available	Available	 Name: Operation_1 Type: ● Pick ○ Place Associated Conveyor or Indexed WA: Conveyor_1 Camera_1 Select Flow: Default Select Hotspot: Hotspot_0 Select Object: Item_1 Object Generation Distance[mm]: 300.00 Trigger Distance[mm]: 300.00 xx2200002001
Conveyor	**Vision/External Sensor**	**I/O**	Available	Unavailable	 Name: Operation_1 Type: ● Pick ○ Place Associated Conveyor or Indexed WA: Conveyor_1 Select Flow: Default Select Hotspot: Hotspot_0 Select Object: Item_1 Object Generation Distance[mm]: 300.00 xx2200002002
Conveyor	**Predefine**	**I/O**			
Conveyor	**Predefine**	**Distance**	Available	Disabled	 Name: Operation_1 Type: ● Pick ○ Place Associated Conveyor or Indexed WA: Conveyor_1 Select Flow: Default Select Hotspot: Hotspot_0 Select Object: Item_1 Object Generation Distance[mm]: 300.00 Trigger Distance[mm]: 300.00 xx2200002003
Indexed work area	**Vision/External Sensor**	**Distance**	Unavailable		 Name: Operation_1 Type: ● Pick ○ Place Associated Conveyor or Indexed WA: IndexedWorkArea_1 Select Flow: Default Select Hotspot: Hotspot_0 Select Object: Item_1 xx2200002004
Indexed work area	**Vision/External Sensor**	**Distance**			
Indexed work area	**Predefine**	**I/O**			
Indexed work area	**Predefine**	**I/O**			

Continues on next page

Filter Setting

	Description
Position Filter Distance	The position filter defines the minimum allowed distance between the different item positions found by a camera or an external sensor. For example, if two or more models are used to identify the same object, there might be one hit for each model at almost the same location. If two positions for the same item are closer in either x- or y-direction than the defined minimum item distance, only the position with the highest sort value will be sent to the robot controller. The sort value can be set for each vision model, see *Adding vision model on page 55*. If **Same level only** is selected, the filtering will only be done between item positions with the same inspection level. **Note** The position filter is not used while predefined positions are used.
Overlap Filter Distance	For linear conveyor, items can be identified in two consecutive frames due to the overlap. The models can have a small variation in the pick/place position between these frames. Items that are found in two consecutive frames and whose pick/place position between these two frames does not vary by more than the overlap filter distance will be regarded as one item. The first identified hit is sent to the robot, and any subsequent hit is filtered out.
Overlap Filter Angle	For circular conveyor, items can be identified in two consecutive frames due to the overlap. The models can have a small variation in the pick/place position between these frames. Items that are found in two consecutive frames and whose pick/place position between these two frames does not vary by more than the overlap filter angle will be regarded as one item. The first identified hit is sent to the robot, and any subsequent hit is filtered out. **Note** For circular conveyor, **Overlap Filter Distance** and **Overlap Filter Angle** are both valid. Which one works depends on which filtering condition is more stringent.

Advanced function - User Script

User script is an advanced function for programming user. For detailed information, see *User script on page 90*.

Distribution Setting

By default all positions are sent to the same work area. It is possible to distribute item positions to more than one work area to balance the load between several robots or to guarantee that all positions are accessed.

All positions for a specific item type are distributed to the robots by a single item distributor. There are four types of item distributors.

- Work area: The item positions are handled by a single conveyor or indexed work area.

Continues on next page

- ByPass: The item positions are discarded, that is not handled by any work area. If no distributor is selected for an item type it will be considered as ByPass.
- LB group: The item positions are handled by the work areas included in a load balance group. Aload balance group is a collection of Work area, ByPass, and ATC group distributors. Item positions will be distributed among the work areas in an optimal way to avoid sending two adjacent positions to the same work area.
- ATC group: Positions are handled by the work areas included in an *Adaptive Task Completion* (ATC) group. An ATC group is a collection of ordered work areas that will get the same item positions. The first robot accesses as many positions as possible. The other robots in the ATC group will access any missed positions. If the last work area in the group is a conveyor work area with start and stop it is guaranteed that all positions will be accessed.

To use either load balancing or ATC the work areas must be arranged in the order that they occur after the position source (for example: the camera or sensor).

The work area that triggers the position source is set automatically. When starting a production, the work area for the robot that is first up and running is set to be the trigger work area. If the robot for a trigger work area is stopped, a work area for another robot that is running will be the one that triggers the position source.

The item distribution tree control shows the items for which positions are to be generated. Accepted and rejected items can be distributed differently.

Distribution

	Description
Item distribution	Set the distribution strategy as **Accept** or **Reject** for all available items for this operation. **Note** Make sure that at least there is one group valid distribution setting under **Item distribution Accept** or **Reject** for all available items. Otherwise an error will pop up when this recipe is selected to do the simulation or production. `{0} lacked valid distribution. Please check settings in Recipe -> Operation.`
Available Distributor	Shows the available distributor for this operation.

Continues on next page

Load balance

Item positions that are distributed by a load balance group are divided among the distributors in the group. A load balance group can contain any number of item distributors and a single distributor can appear several times. The ratio between the number of times a single distributor is added and the total number of distributors defines the ratio of the item positions that are sent by that particular distributor. Item positions are arranged to the distributors in the group in an optimal way to avoid adjacent positions to be sent to the same work area.

If *Adaptive Task Completion* is selected, any defined ATC groups will be listed among the available distributors. Additionally, ATC groups can be added to load balance groups. However, to achieve task completion, the load balance group should only contain ATC groups.

	Description
Load Balance Group	Shows the created load balance group.
Available Distributor	Shows the available distributor for this operation.
New LBGroup	Create a load balance group.
Delete Group	Delete a load balance group.
Rebalance strategies checkbox	Set the rebalancing strategies for current recipe. For more information, see *Rebalancing strategies when a robot goes down on page 52*.

Rebalancing strategies when a robot goes down

There are three ways to rebalance item positions if a robot goes down. The item positions can automatically be sent to the running robots. However, sometimes it can be more convenient to keep sending item positions to robots that are not running.

When sending item positions to a robot controller that has paused. For example, caused by a motors off state, the positions will not be lost until they have passed the robot. As soon as the robot is running again it can start picking immediately.

A robot that has been stopped cannot receive any item positions until it is started again. All items that already have been distributed to it will be lost. When the robot is started again, it will have to wait until new positions reaches the robot.

Rebalancing strategies:

- Rebalance strategies checkbox disabled: Item positions will always be distributed as defined in the distribution tree. If a robot is down some item position will be lost.
- Rebalance among running and paused robots (default setting): Item positions will only be sent to work areas with running or paused robots.
- Rebalance among running robots: Item positions will only be sent to work areas with running robots.

For the different robot states, see *Robot states* in PickMaster® Twin - PowerPac Application manual.

The selection of rebalance strategy is important while using load balancing, for example, to minimise production loss.

Continues on next page

ATC

Adaptive Task Completion guarantees the item positions to be accessed by any robot in an ATC group. An ATC group contains ordered work areas and a single work area is allowed to exist once in a group. All item positions distributed to an ATC group are sent to every work area in the group and the positions not accessed by the first work area will be accessed by any of the other work areas. If the last work area is on a conveyor with start and stop it is guaranteed that all item positions will be accessed by one of the robots in the ATC group.

	Description
Adaptive Task Completion Group	Shows the created adaptive task completion group.
Available Distributor	Shows the available distributor for this operation.
New ATCGroup	Create a adaptive task completion group.
Delete Group	Delete a adaptive task completion group.

Procedure

On the PickMaster Recipe Manager ribbon-tab, click **Process**.

Use this procedure to add a recipe:

1. On the ribbon-tab, click **Recipe.**

   The **Recipe** window opens.
2. Click on the **Add Operation** to add a new operation.
3. Click on the **Operation 1** to open the setting window for the operation.
4. Select the operation type as **Pick** or **Place**.
5. If need, click to select the applicable flow in **Select Flow**.
6. Click to select the item in **Available Objects.**
7. Click to select the work area in **Available Work Areas**.
8. In the **Trigger/Filter Setting** tab, define the trigger or filter setting according to your requirements.
9. If need, click to select and configure the **User Script** according to your requirements.
10. In the **Distribution** tab, drag distributors from the **Available distributors** list to the **Distribution** list.

    There can be only one distributor for each item type. If an item type is missing a distributor, it will be regarded as ByPass.
11. If using load balancing, in the **Load balance** tab, drag a distributor from the **Available distributors** list to a group in the list **Load balance groups.**

    To create a new load balance group, double-click <**New LbGroup**> in the **Available distributors** list.

    Select rebalancing strategy.
12. If using Adaptive Task Completion, in the **ATC** tab, drag a work area from the **Available work areas** list to the **Adaptive Task Completion groups** list.
13. Click **OK.**

    The window is closed.

Continues on next page

Redistributing items from one robot to downstream robots

It is possible to modify the distribution of alredy distributed item positions when they enter a conveyor work area of a robot. The Rapid program, that controls the robot, based on current flow conditions decides to skip an item position and change the type of it. As a result, PickMaster Recipe Manager will redistribute the item position to downstream robots according to the configured distribution strategy for the selected item type.

Modifying a recipe

Managing recipe

Right-click on a **Recipe** in the tree view to manage the recipe.

	Description
Setting	Manage the settings of the selected recipe. When you select **Setting**, the **Recipe Setting** window is displayed. More details about managing a selected recipe is available in the section *Adding a recipe on page 42*.
Delete	Delete the selected recipe.
Rename	Change the name of the selected recipe.
Copy	Create a copy file of the selected recipe with all settings.

4.4 How to add vision model for item or container

4.4.1 Adding vision model

4.4.1.1 Vision modeling

Introduction to vision modeling

There are three different tools available for generating models in a solution. The three tools are:

- *Geometric* which is a pattern recognition tool. See *Configuring a geometric model on page 58*.
- *Blob* which is a detection of two-dimensional shapes within images. See *Configuring blob models on page 67*.
- *Inspection tool* (Inspection II) which makes it possible to combine the *Geometric*, Blob, Histogram and Caliper to generate a model. See *Configuring inspection models on page 73*.

Note

Vision modeling can only be created or edited when the software is connected to real Runtime.

Note

You can import vision models from PickMaster 3 solutions and other PickMaster Recipe Manager solutions.

Importing an existing vision model

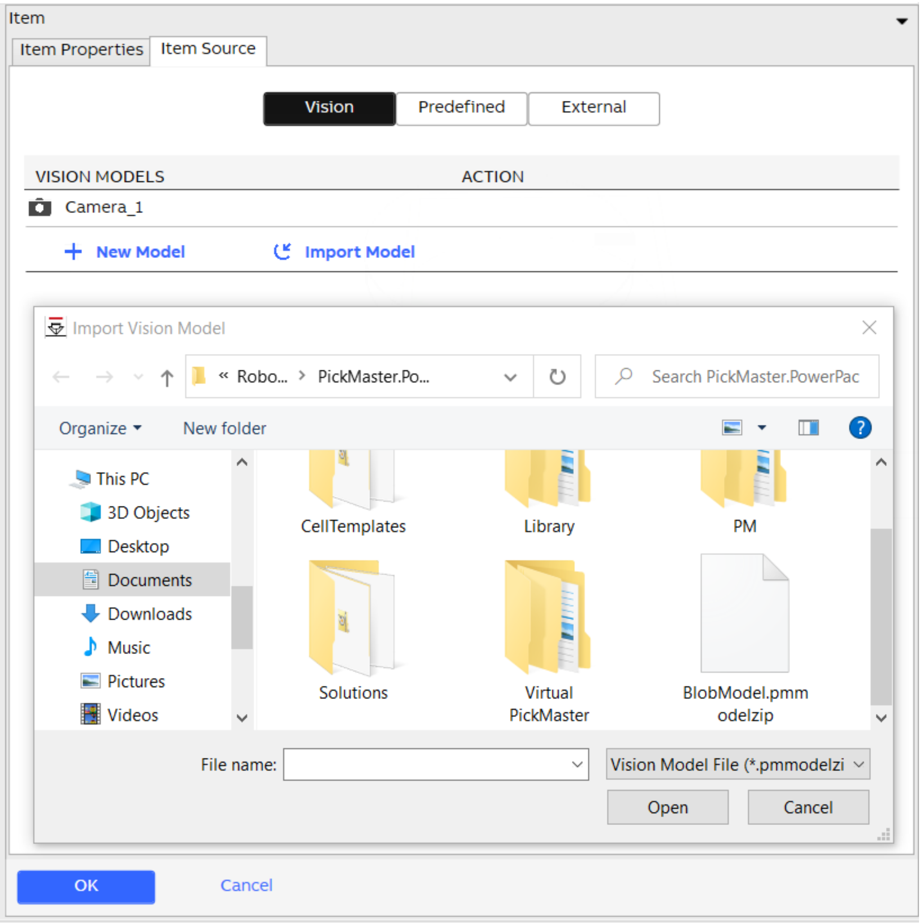
Use this procedure to import an existing vision model.

1. Right-click on one **Item** in the tree view **Process** and select **Setting**.

   The **Item Setting** window is opened.
2. Click to select the **Item Source** tab.
3. In the **Item Source** dialog, click **Import Model** under the required camera.

   The **Import Vision Model** window is opened.
4. Select the valid vision model (`.pmmodel` or `.pmmodelzip`) and click **Open**

Continues on next page



xx2200000479

5 Click **OK**.

Classification of items

Items identified by vision models can be classified as either accepted or rejected. These two types can be distributed to different work areas and be given different item type values accessible from the RAPID program. Item classification can be done by *Geometric*, *Blob*, and the *Inspection tool*.

Vision model parameters in item targets

Item targets identified by a vision model can store a selection of upto 5 vision model parameters in the components `Val1`, `Val2`, `Val3`, `Val4`, and `Val5`. These parameters can be accessed in the RAPID program.

Item targets identified by an *inspection model* can store a selection of parameters from the alignment model and from the included subinspection models.

For each kind of vision model, a *target storage* can be selected for some vision parameters.

External vision models

This function is reserved for next version.

Continues on next page

Related information

Configuring a geometric model on page 58.

Configuring blob models on page 67.

Configuring inspection models on page 73.

4.4.1.2 Configuring a geometric model

Introduction to the geometric model

Geometric is a pattern location search technology. This tool measures:

- Position of the pattern.
- Size relative to the originally trained pattern.
- Angle relative to the originally trained pattern.

Geometric differs from other pattern location technologies as it is not based on pixel grid representations that cannot be efficiently and accurately rotated or scaled. Instead, *Geometric* uses a feature based representation that can be transformed quickly and accurately for pattern matching.

When creating a pattern the following things should be considered.

- Select a representative pattern with consistent features. Reduce needless features and image noise. Train only important features. If necessary, export the image and use an external program to erase noise.
- Larger patterns will provide greater accuracy because they contain more boundary points to resolve at run-time.
- High frequency features are more significant at the outer edges of the pattern.

Models can be classified with the function *Inspection I*. A model can either be defined as accepted or rejected, see *Item Properties tab on page 27*.

To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. See *Using color vision on page 82*.

There are several parameters that can be adjusted to make an efficient model. The configuration is done in the **Geometric Model** tab view and the result is displayed in the **Search Result** window and the **Image Dialog**.

Algorithms

The PickMaster Recipe Manager geometric model supports two pattern-location algorithms:

- PatMax
- PatQuick

PatMax offers higher accuracy than PatQuick, but PatMax requires more time to execute. The PatMax algorithm can also return additional score information.

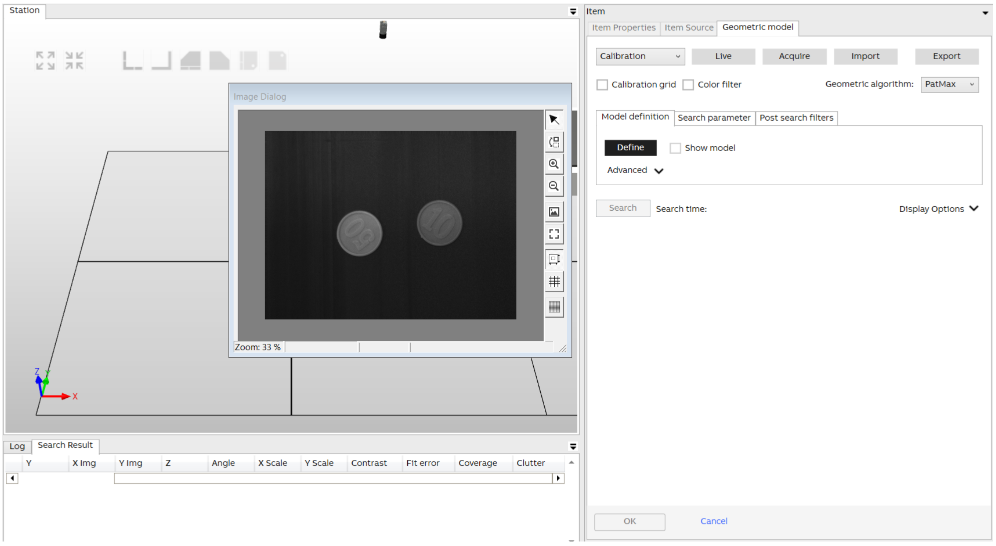
The available configuration parameter is different between PatMax and PatQuick.

Algorithms	Available configuration parameter
PatMax	All configuration parameters

Continues on next page

Algorithms	Available configuration parameter
PatQuick	The configuration parameters EXCEPT: • **Search parameters -> Score Using Clutter** • **Post search filters -> Fit Error** • **Post search filters -> Coverage** • **Post search filters -> Clutter** • **Post search filters -> Target storage** • **Display Options -> Match info** When PatQuick algorithm is selected, the unavailable parameters will be blocked as default value.

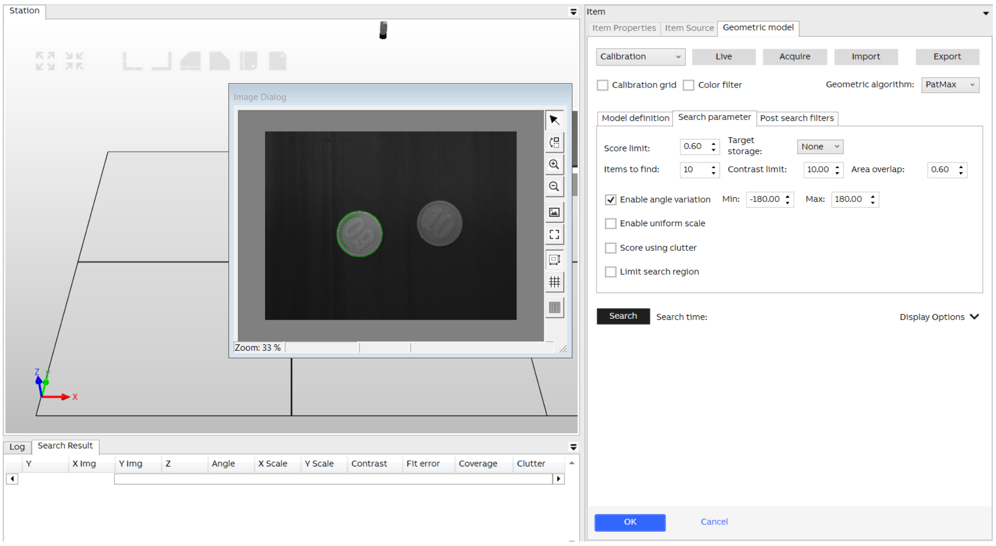
Illustration geometric model Configuration



☐ Calibration grid ☐ Color filter Geometric algorithm: PatMax

☐ Show model

xx2100001647



☐ Calibration grid ☐ Color filter Geometric algorithm: PatMax

Score limit: 0.60 Target storage: None

Items to find: 10 Contrast limit: 10.00 Area overlap: 0.60

☑ Enable angle variation Min: -180.00 Max: 180.00

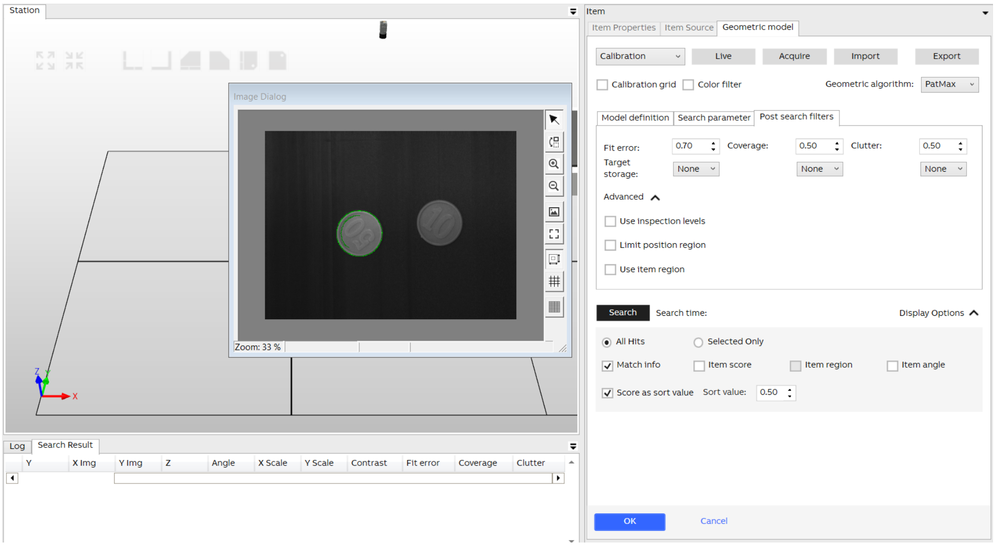
☐ Enable uniform scale

☐ Score using clutter

☐ Limit search region

xx2100001648

Continues on next page



□ Calibration grid □ Color filter Geometric algorithm: PatMax

Fit error: 0.70 Coverage: 0.50 Clutter: 0.50

Target storage: None None None

□ Use inspection levels

□ Limit position region

□ Use item region

● All Hits ○ Selected Only

☑ Match info □ Item score □ Item region □ Item angle

☑ Score as sort value Sort value: 0.50

xx2100001649

Configuring a geometric model

Use this procedure to configure a geometric model.

1. Right-click on one **Item** in the tree view **Process** and select **Setting**.

   The **Item Setting** window is opened.
2. Click to select the **Item Source** tab.
3. In the **Item Source** dialog, click **New model** under the required camera and select **Geometric**.

   The **Image Dialog** and **Geometric** dialog are opened.
4. In the **Geometric model**, click **Live**, **Acquire**, or **Import** to get an image. Select the calibration that has set in the **Camera Calibration** from the **Calibration** list.

   Select the preferred algorithm for this model in the **Geometric algorithm**. For more details about the available algorithms, see *Algorithms on page 58*.

   Select the **Calibration grid** checkbox to display help lines for the coordinate system. The help lines can be moved with the mouse to make it easier to train a pattern.
5. If color filtering should be used select the **Color filter** checkbox to enable the filter. Configure the filter parameter in the **Color Filter** tab. See *Using color vision on page 82*.
6. In the **Model definition** part, define a model for the pattern using an image in front of the camera or using an imported image. The selected calibration will be used.

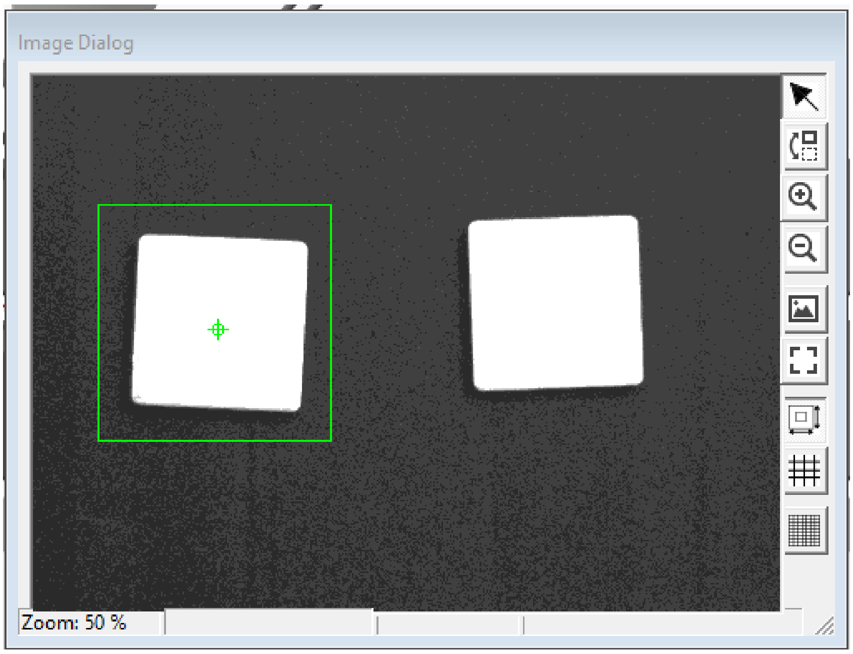
Continues on next page

Note

When importing a vision model it is required to enter model configuration and re-select which calibration to use from the calibration drop-down menu. This is required even if there is only one calibration defined. If this is not performed then further actions may produce the error `No valid calibration for the` *`Geometric`* `model`.

a If the height of the item is to be defined, choose an appropriate calculation method before training the item. **Model Height** is used as the basic height for the trained item. **Pick Offset** is used to make up the deviation of the picking point with this calculation method. For more information, see *Working with products of varying height (2.5D vision) on page 110*.

b Click **Define** to define a model. Drag the rectangle so it covers the pattern and move the cross to the desired pick/place position. To maintain the greatest accuracy, the pick/place position should be placed close to the center of the trained pattern.



xx1800001708

c Click **Train** to train the pattern.

d Select **Show Model** to show the features of the trained models in the search image.

e If needed, click **Advanced** to access more model settings.

f Click **Adjust Granularity** to define the levels in the **Fine** and **Coarse** boxes. Granularity is a radius of influence, in pixels, which determines the detection of a feature in a pattern. *Geometric* locates patterns in the search image by first searching only for large features. After locating one or more pattern instances, it uses smaller features to determine

Continues on next page

the precise transformation between the trained pattern and the pattern in the search image. *Geometric* uses the same range of granularity that is computed when training the pattern to detect features in the search image. The granularity parameters *fine* and *coarse* are auto-selected when training the pattern and often these values are the best. These can also be set manually. The lower limit is 1 and upper limit is 25.5.

g Select **Ignore polarity** to ignore if the features are dark on bright or bright on dark.

> **Note**
>
> *Geometric* will not care if a product is light on a dark background or dark on a light background. This is useful when the background is, for example, a grid.

h Increase the value of **Elasticity** to allow for any expected non-linear shape distortion, for example, for organic products and so on. The value represents the maximum distance between a trained feature and a matched feature in pixels. The lower limit is 0 and upper limit is 25. This setting is useful for products of varying shape.

7 In the **Search parameters** part, set parameters to limit the search procedure and the analysis time.

Score Limit indicates how closely the found item matches the trained model. A score of 1 indicates a perfect match while a score of 0 indicates that the pattern does not match at all. The higher a score threshold is defined the faster *Geometric* will be able to perform a search.

Target Storage indicates the variables in Rapid. For more information, see *GetItmTgt - Get the next item target* in PickMaster® Twin - PowerPac Application manual.

Items to Find is the number of items that is expected to be present in the image. If there are more items present in the image these will not be reported by *Geometric*.

Contrast Limit defines the minimum image contrast of each item that is found in the image. The contrast is the average difference in gray-level values for all of the boundary points that *Geometric* matched between the trained model and the found item in the search image. *Geometric* considers only items with a contrast value that exceeds the contrast limit.

Area Overlap defines how much multiple patterns in the image are allowed to largely overlap each other. *Geometric* assumes that these patterns actually represent the same item in the image. When two patterns overlap by a percentage greater than the area overlap threshold they are treated as a single pattern.

Enable Angle variation defines the acceptable rotation for the items. If an item has a rotation outside the valid range it will be discarded by the vision system. Default +/- 180 degrees.

Continues on next page

Enable Uniform Scale is a threshold that accepts hits that differ in size relative to the taught vision model. A scale value of 1 indicates that there are no differences between the found item and the taught vision model. A value <1 indicates a smaller model.

Score Using Clutter[I] defines a measure of the extent to which the found item contains features that are not present in the trained vision model. By default the *Geometric* analysis ignores clutter when scoring which means that the patterns receive the same score regardless of the presence of extra features. If this checkbox is selected, clutter is included in the calculation of the score. If the application is an alignment application in which the background does not change, **Score Using Clutter** should be selected.

Limit Search Region limits the search area for the *Geometric* analysis. Only objects within this area will be found. A smaller search area will decrease the search time.

Note

When combining **Fine**/**Coarse Granularity** and **Uniform Scale** a slight difference in the score can appear between design time and running time. Therefore, the model should be tested in running time to verify that items are identified as expected.

8 In the **Post search filters** part, define the score values for each pattern in the search image.

Fit Error is a measure of the variance between the shape of the trained pattern and the shape of the pattern found in the search image. If the found pattern in the search image is a perfect fit for the trained pattern, the fit error is 0. This is only valid for models with PatMax algorithm.

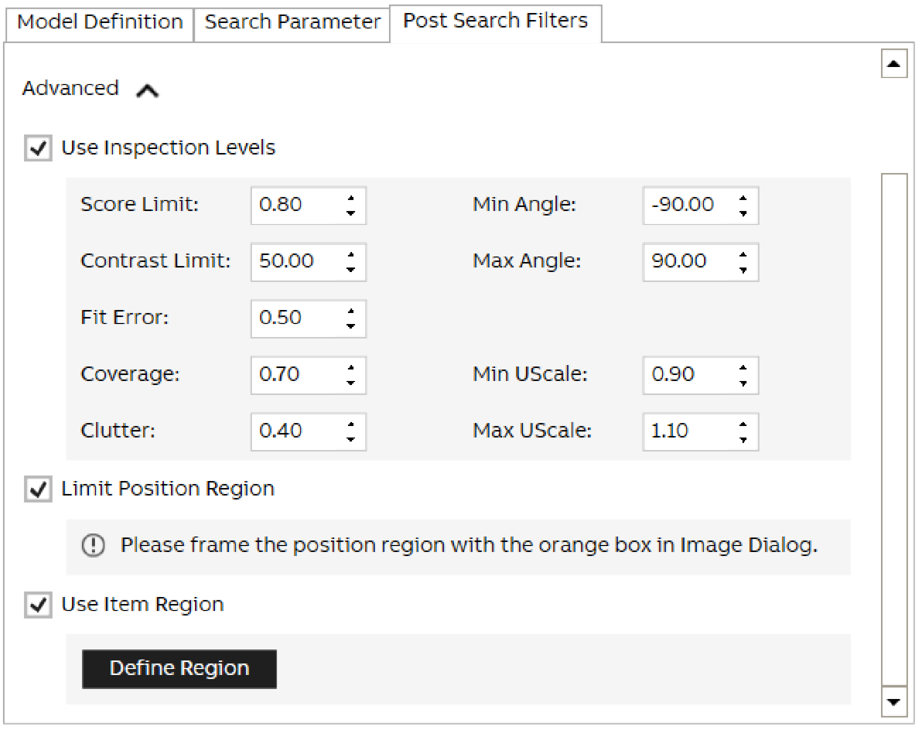
Coverage is a measure of the extent to which all parts of the trained pattern are also present in the search image. If the entire trained pattern is also present in the search image, the coverage score is 1. Lower coverage scores indicate that less of the pattern is present. This parameter can be used to detect missing features. This is only valid for models with PatMax algorithm.

Clutter is a measure of the extent to which the found pattern contains features that are not present in the trained pattern. A clutter of 0 indicates that the found pattern contains no extra features. A clutter score of 1 indicates that for every feature in the trained pattern there is an additional extra feature in the found pattern. The clutter can exceed 1.0. This is only valid for models with PatMax algorithm.

Target Storage indicates the variables in Rapid. For more information, see *GetItmTgt - Get the next item target* in PickMaster® Twin - PowerPac Application manual. This is only valid for models with PatMax algorithm.

If more settings are required, click **Advanced** to open the **Advanced Search Settings** dialog where the following settings are found:

Continues on next page



☑ Use Inspection Levels

Score Limit:	0.80	Min Angle:	-90.00
Contrast Limit:	50.00	Max Angle:	90.00
Fit Error:	0.50		
Coverage:	0.70	Min UScale:	0.90
Clutter:	0.40	Max UScale:	1.10

☑ Limit Position Region

Please frame the position region with the orange box in Image Dialog.

☑ Use Item Region

xx2100002333

Use Inspection Levels - Inspection I, this inspection is also called *Inspection I* in PickMaster Recipe Manager. With this function it is possible to classify the found models into two categories. A model can either be classified as accepted or rejected. An accepted model has better search results than the rejected model. The item type number is defined for the accepted and rejected model in the **Item** dialog, see *Item Properties tab on page 27*. The item type can be read in the RAPID code, see *RAPID programs included in installation* in PickMaster® Twin - PowerPac Application manual.

In the Inspection parameter section, all models that fulfill the conditions specified for the search parameters and the post filters will be classified. Select **Use Inspection Levels** to define the parameter that will divide the found items into the two categories. If **Use inspection levels** is not selected all found models are classified as an accepted model.

For **Score**, **Contrast**, and **Coverage**, items with a value larger than the defined value in **Inspection Parameter** will be defined as accepted.

For **Angle** and **Uniform Scale**, items with a value between the defined values in **Inspection Parameter** will be defined as accepted.

For **Fit Error** and **Clutter** a value less than the defined one will be classified as accepted.

Limit Position Region defines if the *Geometric* analysis is done on the whole image. Objects found within this area will be handled as normal. Object found outside this area will be discarded.

To define an item region, select **Use Item Region** checkbox and click **Define Region**. Adjust the polygon showed around the found object using vertices. Then click **Train**. The polygon can have 2 to 16 vertices.

Continues on next page

9 In the **Display options** part, select the type of information to display in graphics.

Match Info displays the quality of the matched boundary points in the search image. This is only valid for models with PatMax algorithm. Boundary points drawn in:

- Red are poor matches.
- Yellow are fair matches.
- Green are good matches.

Item Score displays the score for the selected item in the image window.

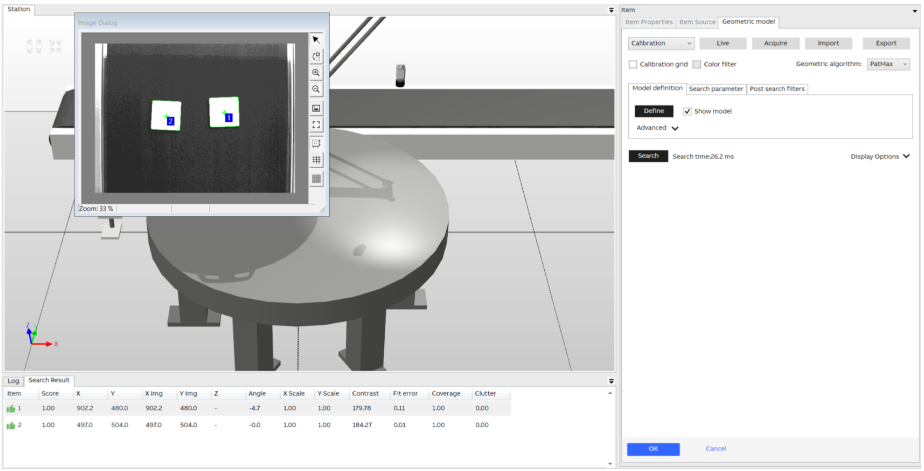
Item Region displays the regions in the image window. Red regions indicate an overlap and the corresponding hits will be considered as discarded.

Item Angle displays the angle of the item that will be sent to the robot. This angle is relative to the trained model.

Sort value is used if there is more than one hit for the same item. Only the hit with the highest sort value will be sent to the robot controller. The sort value can be set individually for all models or the *Geometric* score can be used by selecting **Score as sort value.**

10 Click **Search** to analyze the image. If needed, define sort value.

The result is displayed as an image with numbered hits in the **Image** dialog, and a corresponding result list.



□ Calibration grid □ Color filter

Geometric algorithm: PatMax

☑ Show model

Search time:26.2 ms

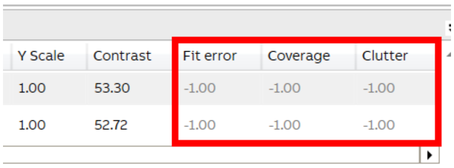
Item	Score	X	Y	X Img	Y Img	Z	Angle	X Scale	Y Scale	Contrast	Fit error	Coverage	Clutter
1	1.00	902.2	480.0	902.2	480.0	-	-4.7	1.00	1.00	179.78	0.11	1.00	0.00
2	1.00	497.0	504.0	497.0	504.0	-	-0.0	1.00	1.00	184.27	0.01	1.00	0.00

xx2100001636

Continues on next page

Note

When the algorithm is selected as PatQuick, the unavailable parameters (**Fit error**, **Coverage** and **Clutter**) in result list will display as below.



Y Scale	Contrast	Fit error	Coverage	Clutter
1.00	53.30	-1.00	-1.00	-1.00
1.00	52.72	-1.00	-1.00	-1.00

xx2400001696

Model hits are normally classified as accepted. If inspection is used, hits can be classified as either accepted or rejected. See *Item Properties tab on page 27*. Hits that do not fulfill all the requirements or hits with overlapping regions will not be accessed by any robot and are classified as discarded. The hits shown in the result list are marked with an icon identifying its classification. For hits that are not accepted, the parameter that failed is marked with either red or blue in the result list.

Search Time displays the time it takes to analyze the image in ms.

11 Click **OK**.

[i] Only available for models with PatMax algorithm.

Note

Items located after a search operation in the *Geometric* configuration window is presented as discarded due to item region overlap even if they are actually rejected due to another parameter (fit error, clutter, and so on). This happens only if the item region is activated and the item regions overlap with each other in running time. However, the discarded items are removed before applying the item region.

Geometric parameters in item targets

The *Geometric* parameters `Score`, `fit error`[i], `coverage`[i], and `clutter`[i] can be selected for the target storage.

4.4.1.3 Configuring blob models

Introduction to blob models

The simplest kinds of images that can be used for machine vision are two-dimensional shapes or blobs. Blob analysis is the detection of two-dimensional shapes within images. It finds objects by identifying groups of pixels that fall into a predefined grayscale range.

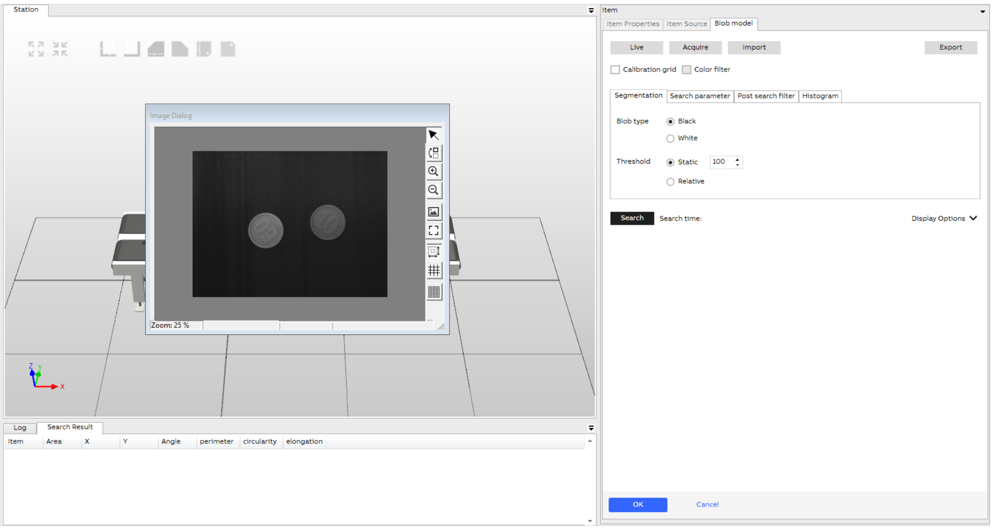
This kind of analysis is well suited for applications where:

- Objects vary much in size, shape, and/or orientation.
- Objects are of a distinct shade of gray not found in the background.

Blob analysis works best with images that can be easily segmented into foreground and background pixels. Typically, strong lighting of scenes with opaque objects of interest produces images suitable for an analysis like this.

To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. See *Using color vision on page 82*.

Illustration Blob Configuration



□ Calibration grid □ Color filter

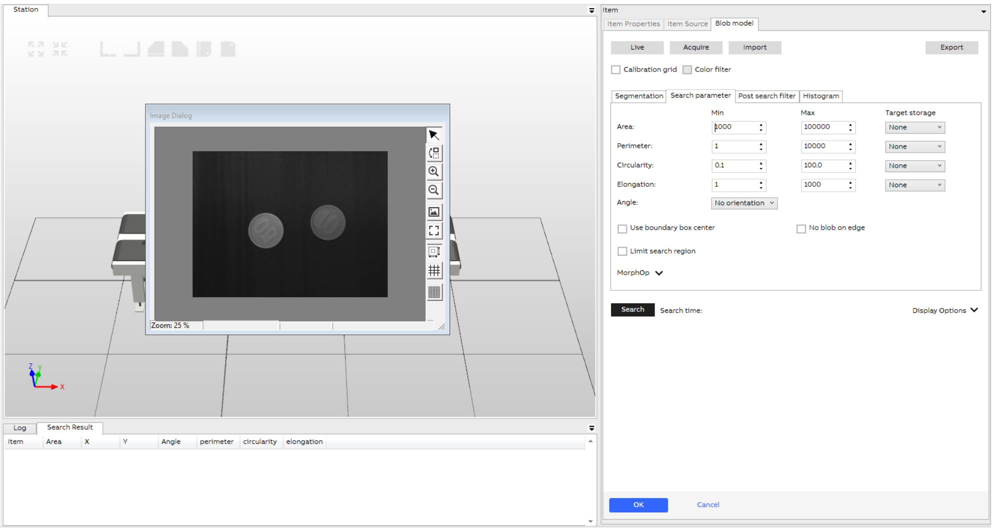
Blob type ● Black ○ White

Threshold ● Static 100 ○ Relative

Item	Area	X	Y	Angle	perimeter	circularity	elongation

xx2100001661

Continues on next page



□ Calibration grid □ Color filter

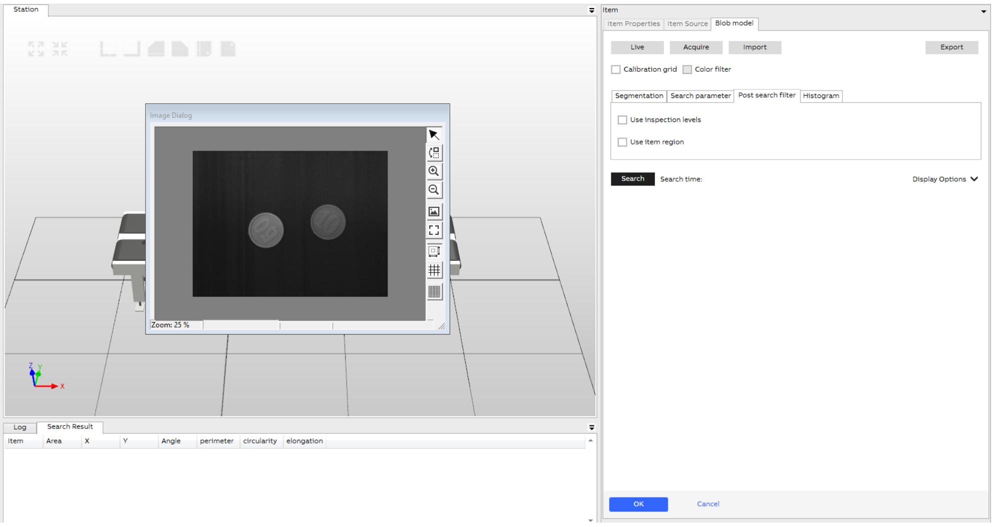
	Min	Max	Target storage
Area:	1000	100000	None
Perimeter:	1	10000	None
Circularity:	0.1	100.0	None
Elongation:	1	1000	None
Angle:	No orientation		

□ Use boundary box center □ No blob on edge

□ Limit search region

Search time:

xx2100001662



□ Calibration grid □ Color filter

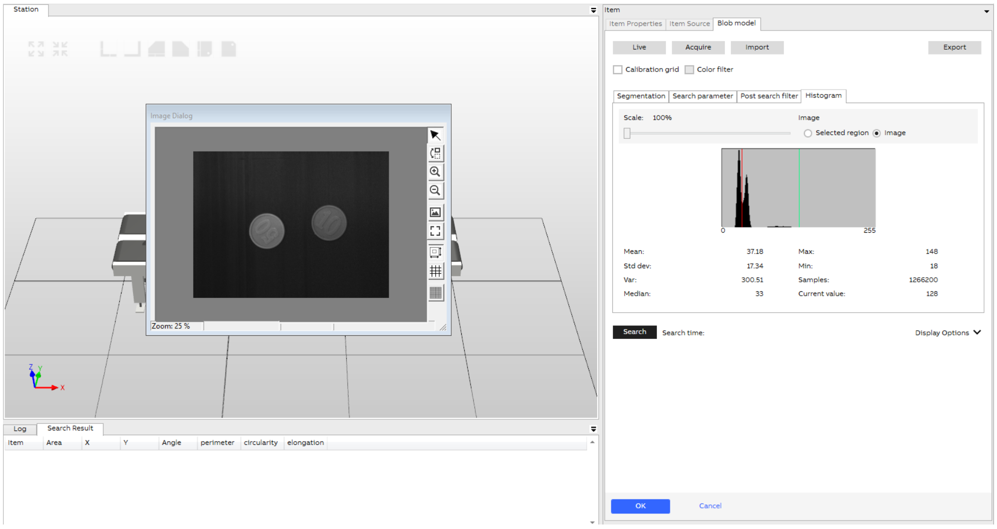
□ Use inspection levels

□ Use item region

Search time:

xx2100001663

Continues on next page



Scale: 100%

Mean:	37.18	Max:	148
Std dev:	17.34	Min:	18
Var:	300.51	Samples:	1266200
Median:	33	Current value:	128

xx2100001664

Configuring a blob vision model

Use this procedure to configure a blob vision model.

1. Right-click on one **Item** in the tree view **Process** and select **Setting**.

   The **Item Setting** window is opened.
2. Click to select the **Item Source** tab.
3. In the **Item Source** dialog, click **New model** and select **Blob**.
4. In the **Image** part, click **Live**, **Acquire**, or **Import** to get an image. Select the **Calibration origin** checkbox to display help lines for the coordinate system. Click **Histogram** to display a graph of the pixel distribution in the acquired image.

   If color filtering should be used, select the **Color filter** checkbox to enable the filter and configure the filter parameter in the **Color Filter** tab. See *Using color vision on page 82*.
5. Click to select **White** in the **Segmentation** under **Model Definition**.

   In the **Segmentation** part, select segmentation method and blob type.

   Segmentation is the division of the pixels in an image into object pixels and background pixels. Typically objects are assigned a value of 1 while background pixels are assigned a value of 0.

   Static method uses gray values to divide blob pixels and background pixels. All pixels with a grayscale value below the threshold are assigned as object pixels, while all pixels with values above the threshold are assigned as background pixels.

   Relative method uses a relative threshold expressed as the percentages of the total pixels between the left and right tail to divide blob pixels and background pixels. Tails represent noise-level pixels that lie at the extremes of the histogram (the lowest and the highest values).

   Static is faster than relative segmentation because the gray levels corresponding to the percentages do not have to be computed. Static

Continues on next page

segmentation can test for absence of a feature in a scene, whereas relative segmentation will always find a blob in the scene.

6 Adjust the parameters in the **Search Parameter** according to your requirements.

In the **Search Parameters** part, define the values for the feature.

Area is expressed in mm^2.

Perimeter is expressed in mm.

Circularity defines the circularity. A value of 1 means perfectly circular and completely filled (no holes).

Elongation is the ratio of the feature’s second moment of inertia about its second principal axis to the feature’s second moment of inertia about its first principal axis.

Angle defines how the found item is sent to the controller.

- **No Orientation** means that the found item is sent to the controller with angle 0 (zero).
- **First Principal Axis** means that the found item is sent down with the angle around the first principal axis. The angle is relative to the x-axis and can be ±90 degrees.

Use boundary box center defines if the position of a blob will be at the center of its boundary box instead of at its center of mass.

No Blob On Edge defines if a blob connected to the edge of the search area should be reported.

Use Inspection Levels defines if the found models should be classified. See *Item Properties tab on page 27*. The item type can be read in the RAPID code, see *RAPID programs included in installation* in PickMaster® Twin - PowerPac Application manual. Select **Use Inspection Levels** to open the Inspection Parameters part.

If **Use Inspection Levels** is not selected all found models are classified as accepted. All models that fulfill the conditions specified for the **Search Parameters** will be classified.

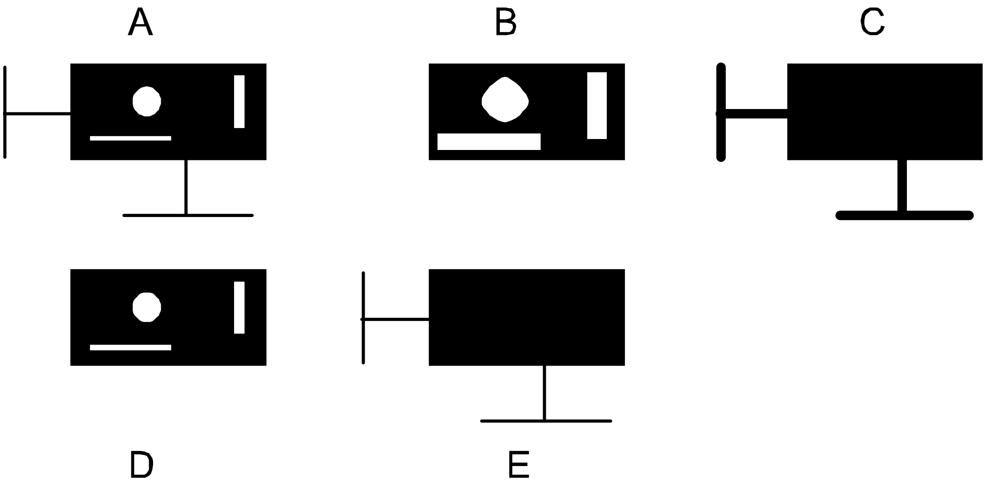
Limit Search Region limits the search area for the blob analysis. Only objects within this area will be found.

Note

Tune the blob tool by pressing **Search** and the blob algorithm lists all the blobs. Adjust the size threshold limit to filter out blobs that are too large or too small. Tune other parameters if necessary.

7 If needed, in the **MorphOp** part, select the **Morphological** and/or **Clean Up** checkboxes and define the settings.

Continues on next page



xx0900000542

A	Original
B	Erosion
C	Dilation
D	Opening
E	Closing

Morphological settings:

- **Erode** reduces or eliminates object features, increases the thickness of holes within an object. This operation replaces each pixel in the image with the maximum value of the pixels and each of its eight vertical and horizontal neighbors.
- **Dilation** reduces or eliminates holes within an object, increases the thickness of an object's features. This operation replaces each pixel in the image with the minimum value of the pixel and each of its eight vertical and horizontal neighbors.
- **Closing** eliminates holes. Preserves small features. An erosion operation is applied to the image, followed by a dilation operation.
- **Opening** preserves holes. Eliminates small object features. A dilation operation is applied to the image, followed by an erosion operation.

Clean up settings:

- **Prune** is used to ignore, but not remove features, that are below a specified size (connectivity size). When an image is pruned of all features below a certain size, the blob measures returned for the blob that enclosed the pruned features are computed as though the pruned features still existed, but the pruned features themselves are not counted.
- **Fill** is used to fill in pruned features with gray values from neighboring pixels on the left. The pixels value that is used to fill the feature is the value of the pixel to the immediate left of the feature being filled. As each row of pixels in the feature is filled, the pixel value to the immediate left of that row of pixels is used as the fill value for that row.

Continues on next page

- **Connectivity** defines the minimum size (in pixels) that a blob can have to be considered. Is used with either prune or fill.

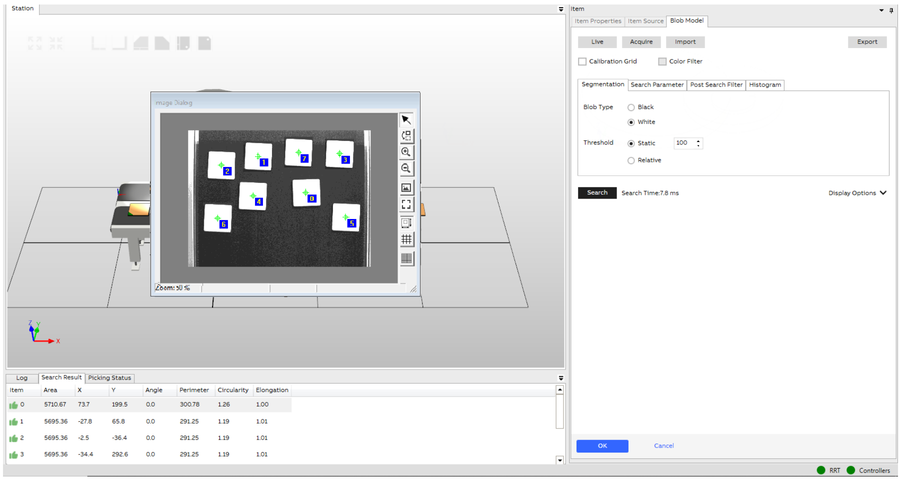
8 In the **Item region** part, select the **Use Item Region** checkbox and click **Define Region**. Adjust the polygon showed around the found object using vertices. Then click **Train**.

The polygon can have 2 to 16 vertices.

9 Click **Search** in the **Display Options**.

Tip

If the search result matches with the **Image Dialog**, the configuration succeeds.



□ Calibration Grid □ Color Filter

Blob Type ○ Black ● White

Threshold ● Static 100 ○ Relative

Search Time:7.8 ms

Item	Area	X	Y	Angle	Perimeter	Circularity	Elongation
0	5710.67	73.7	199.5	0.0	300.78	1.26	1.00
1	5695.36	-27.8	65.8	0.0	291.25	1.19	1.01
2	5695.36	-2.5	-36.4	0.0	291.25	1.19	1.01
3	5695.36	-34.4	292.6	0.0	291.25	1.19	1.01

xx2100001637

In the **Display Options** part, select **Segmentation image** to display the processed image. Select how the result will be displayed.

- **Item Area** displays the area of the blob in the image window.
- **Boundary Box** displays the minimum horizontal rectangle that contains the whole blob.
- **Item region** displays the regions in the image window. Red regions indicate an overlap and the corresponding hits will be considered as discarded.
- **Blob angle** displays the angle of the item that will be sent to the robot.
- **Score Value** displays the score for the selected item in the image window.

10 Click **OK**.

Blob parameters in item targets

The blob parameters `Area`, `perimeter`, `circularity`, and `elongation` can be selected for the target storage.

4.4.1.4 Configuring inspection models

Introduction to inspection models

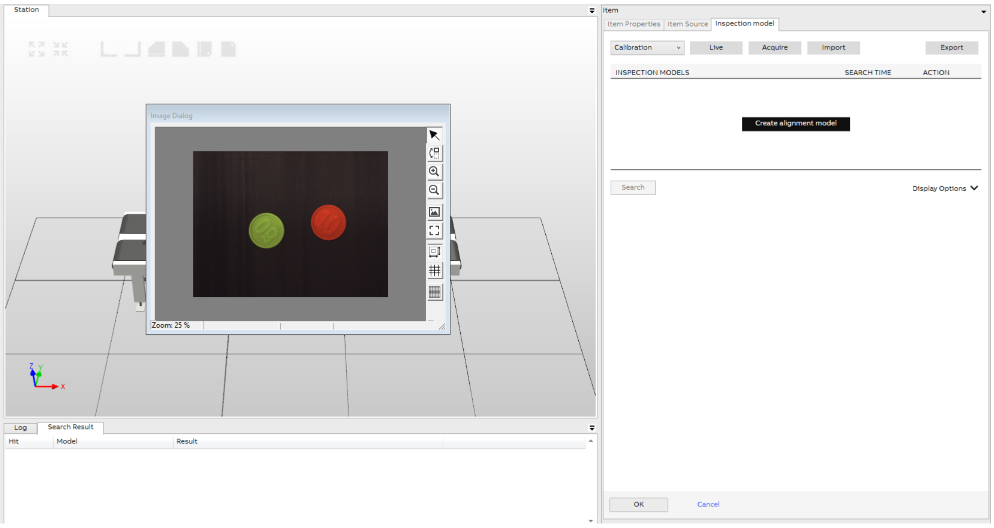
Inspection models make it possible to combine several models of *Geometric*, blob, histogram and *Caliper*. This is sometimes referred to as *Inspection II*.

An inspection model always consists of an alignment model. The alignment model can either be a *Geometric* or blob works as the reference for the inspection model. It is this model's position and rotation that is the pick/place position and rotation for the item.

Inspection areas are defined relative to the alignment model and either blob, histogram, *Caliper* or *Geometric* can be done within each of these areas. Conditions such as number of found items and location relative to the alignment model can be set.

For a found item to be classified as accepted, all inspection areas and the alignment model must be classified as accepted. If one of the inspection areas does not fulfill the given conditions the corresponding item is classified as rejected.

Illustration Inspection Configuration



xx2100001668

Configuring inspection models

Use this procedure to configure inspection models.

1. Right-click on one **Item** in the tree view **Process** and select **Setting**.

   The **Item Setting** window is opened.
2. Click to select the **Item Source** tab.
3. In the **Item Source** dialog, click **New model** and select **Inspection**.
4. In the **Image** part, click **Live**, **Acquire**, or **Import** to get an image.
5. In the **Inspection model** part, define the relationships between the alignment model and its corresponding inspection areas.

   The created models are shown in a tree view.

Continues on next page

Alignment Model defines the position and orientation of any found items. For more information on the alignment model configuration dialog, see *Vision modeling on page 55*.

Sub Inspection Model adds inspection areas to an alignment model. See *Sub inspection models on page 75*.

Edit opens the configuration dialog for the selected model. When an existing alignment model is modified the relations to the inspection areas must be retrained.

Delete is used to delete the selected model and corresponding inspection area.

Edit Area shows the current model's area. The area can be rearranged for the selected sub inspection model.

6 Click **Create Alignment Model** to open the **Select Model Type** drop-down list.

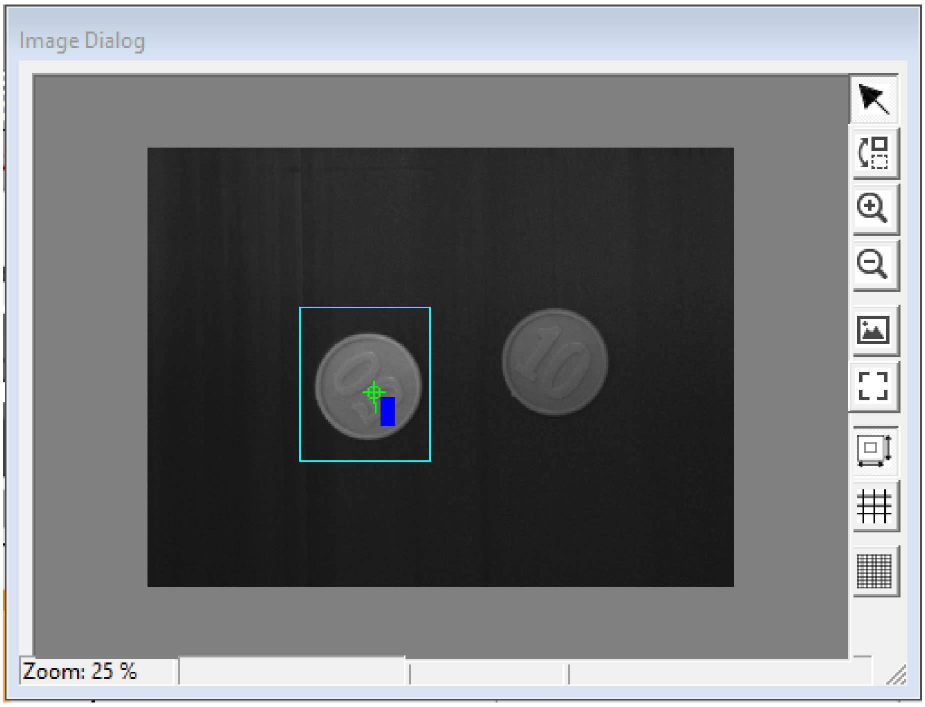
7 Select **Geometric** or **Blob** in the drop-down list to create the alignment model. For detail procedures on how to create a geometric model or a blob model, see *Configuring a geometric model on page 58* or *Configuring blob models on page 67*.

8 Click **+ Sub Inspection Model** to open the **Select Model Type** drop-down list.

9 Select **Geometric**, **Blob**, **Histogram** or **Caliper** in the drop-down list to create the sub model.

10 Click **OK** on the popped-up dialog to edit area.

11 Drag the rectangle so it covers the pattern.



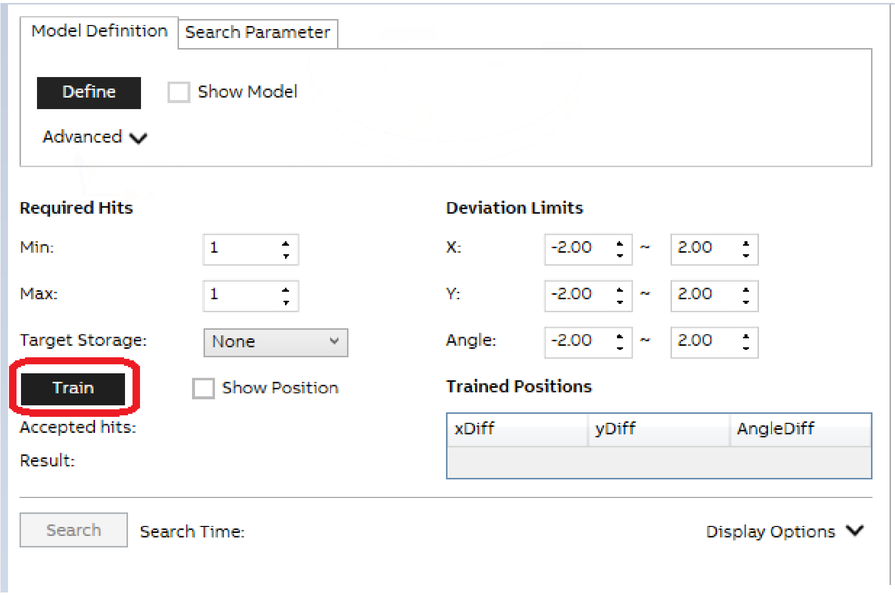
xx2100002275

Continues on next page

12 Click **Edit** button to open the corresponding model creating window. For detail procedures on how to create a Geometric, Blob, Histogram or Caliper model, see *Configuring a geometric model on page 58 Configuring blob models on page 67*, *Histogram on page 76* and *Caliper on page 79*.

Note

For Geometric sub model, after **Define** and **Train** the models, another **Train** need to be done.



Required Hits	
Min:	1
Max:	1
Target Storage:	None

Deviation Limits			
X:	-2.00	~	2.00
Y:	-2.00	~	2.00
Angle:	-2.00	~	2.00

xDiff	yDiff	AngleDiff

xx2100002277

13 Click **Search**.

The result is displayed as an image with numbered hits in the **Image Dialog** and a corresponding detailed list in the **Search Result** window.

Tip

If the search result matches with the **Image Dialog**, the configuration succeeds.

14 Click **OK**.

Sub inspection models

Introduction

Sub inspection models are used to add inspection areas to an alignment model. Each area uses a specified sub inspection model. The inspection area defines where the sub model is to perform its analysis relative to the alignment model. The areas are shown in the image and should be moved and resized to cover the area to analyze.

Continues on next page

Sub inspection models are configured in their own dialogs. When testing a sub inspection model the alignment hit is shown in the image window together with the corresponding inspection area. Sub inspection models only analyze the part of the image defined by its inspection area.

Geometric

A geometric sub inspection model is configured in the same way as a *PatMax* model. See *Configuring a geometric model on page 58*. In addition, the relative positions of the found items and the corresponding alignment hit must be trained.

Required hits defines the number of hits with the sub inspection model within the inspection area that are required for the result to be considered as accepted.

Deviation limits defines the allowed deviations from the trained positions.

After a search and the items are found within the inspection area their positions must be trained. The relative positions are listed as **xDiff**, **yDiff**, and **AngleDiff**.

Click **Train** to save the positions of the found items relative to the alignment hit.

Geometric subinspection parameters in item targets

The parameter `Number of hits` can be selected for the target storage.

Blob

A blob sub inspection model is configured in the same way as a blob model. See *Configuring blob models on page 67*. In addition, the number of required hits must be configured.

Required hits defines the number of hits with the sub inspection model within the inspection area that are required for the result to be considered as accepted.

Blob subinspection parameters in item targets

The parameter `Number of hits` can be selected for the target storage.

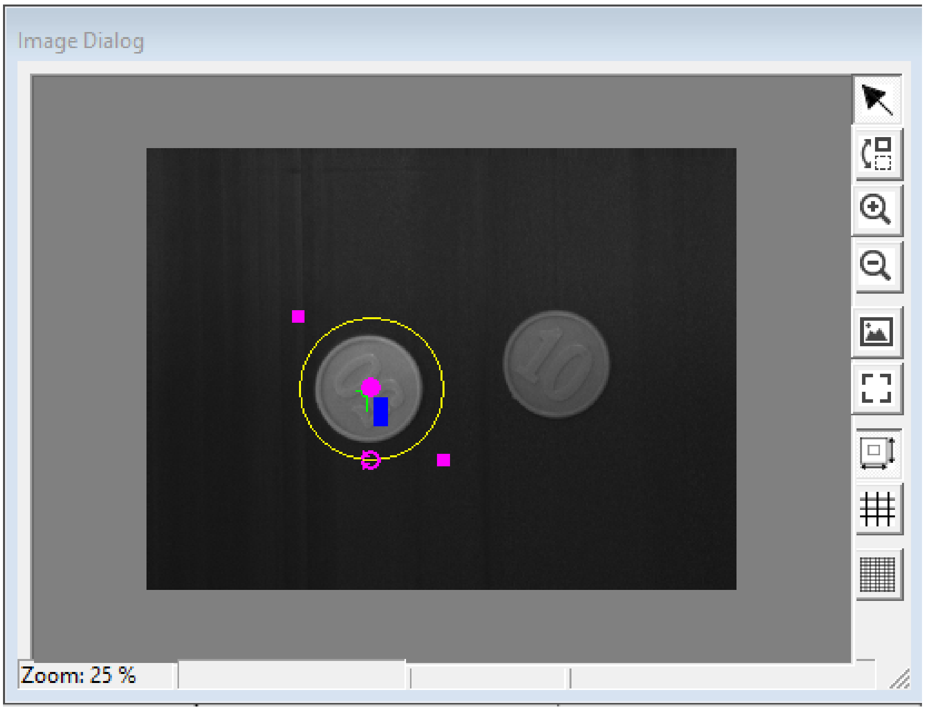
Histogram

The histogram tool measures the color or the gray level within any given area. While using a monochrome camera the histogram tool measures the gray level within a given area. Similarly, if a color camera is used each of the three color channels (Red, Green, and Blue) is measured separately. The histogram tool is useful when the objects to be identified and classified have similar shapes but different colors.

The inspection area for a histogram sub inspection model is graphically represented as a circle. But the area used in the histogram analysis is actually a square aligned with the image but enclosed by the inspection area.

1. Click **+ Sub Inspection Model** to open the **Select Model Type** drop-down list.
2. Select **Histogram** in the drop-down list to create the sub model.
3. Click **OK** on the popped-up dialog to edit area.
4. Drag the circle so it covers the pattern.

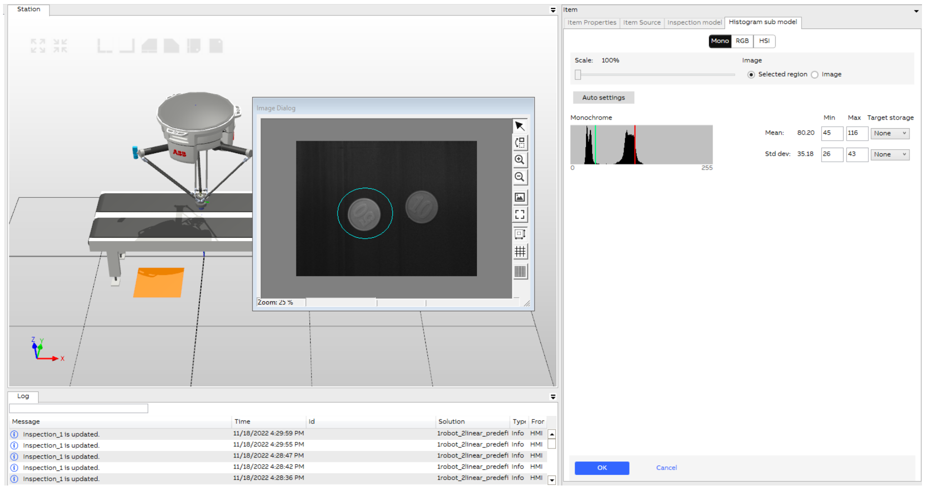
Continues on next page



xx2200001129

5 Click **Edit** icon under **Action** to open the histogram model editing window.

6 Press **Auto Settings** to automatically get an appropriate range limits(**Min.** and **Max.** values) for the histogram. Alternatively, the **Min.** amd **Max.** values can be set manually by sliding the red and green bars across the histogram or by simply entering values into the text boxes. For a product to be accepted, both the standard deviation and the mean value have to be within the specified limits. When using color vision the histograms for all channels must fall within the limits.



Scale: 100%

Image: ● Selected region ○ Image

		Min	Max	Target storage
Mean:	80.20	45	116	None
Std dev:	35.18	26	43	None

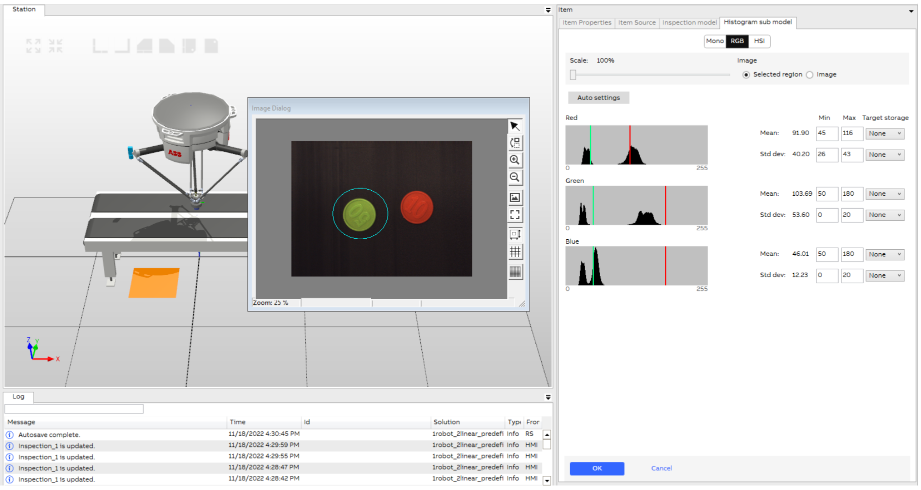
Log

```
Message                     Time                   Id   Solution                Type  From
Inspection_1 is updated.    11/18/2022 4:29:59 PM       1robot_2linear_predefi  Info  HMI
Inspection_1 is updated.    11/18/2022 4:29:55 PM       1robot_2linear_predefi  Info  HMI
Inspection_1 is updated.    11/18/2022 4:28:47 PM       1robot_2linear_predefi  Info  HMI
Inspection_1 is updated.    11/18/2022 4:28:42 PM       1robot_2linear_predefi  Info  HMI
Inspection_1 is updated.    11/18/2022 4:28:36 PM       1robot_2linear_predefi  Info  HMI
```

xx2200001126

7 If change to Tab **RGB** or **HSI**, the window for the colors will show up.

Continues on next page



Scale: 100%

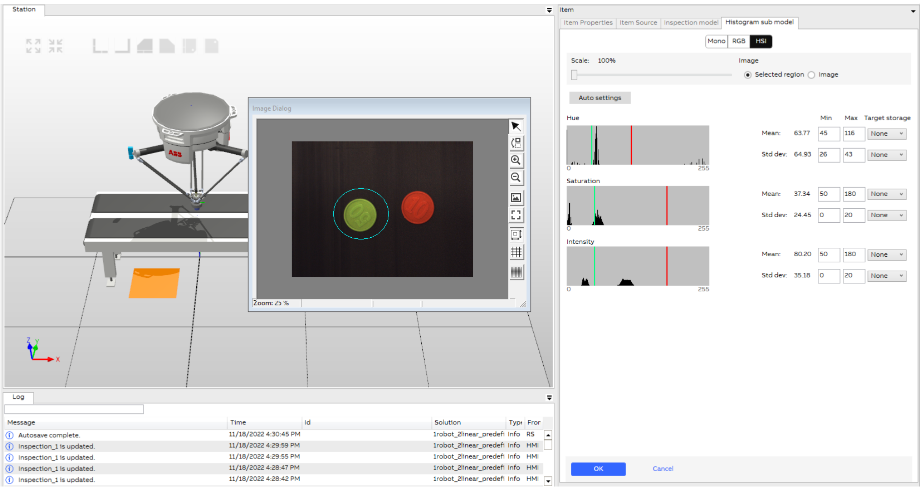
Image: ● Selected region ○ Image

Zoom: 25 %

			Min	Max	Target storage
Red	Mean:	91.90	45	116	None
	Std dev:	40.20	26	43	None
Green	Mean:	103.69	50	180	None
	Std dev:	53.60	0	20	None
Blue	Mean:	46.01	50	180	None
	Std dev:	12.23	0	20	None

Message	Time	Id	Solution	Type	Fror
Autosave complete.	11/18/2022 4:30:45 PM		1robot_2linear_predefi	Info	RS
Inspection_1 is updated.	11/18/2022 4:29:59 PM		1robot_2linear_predefi	Info	HMI
Inspection_1 is updated.	11/18/2022 4:29:55 PM		1robot_2linear_predefi	Info	HMI
Inspection_1 is updated.	11/18/2022 4:28:47 PM		1robot_2linear_predefi	Info	HMI
Inspection_1 is updated.	11/18/2022 4:28:42 PM		1robot_2linear_predefi	Info	HMI

xx2200001127



Scale: 100%

Image: ● Selected region ○ Image

Zoom: 25 %

			Min	Max	Target storage
Hue	Mean:	63.77	45	116	None
	Std dev:	64.93	26	43	None
Saturation	Mean:	37.34	50	180	None
	Std dev:	24.45	0	20	None
Intensity	Mean:	80.20	50	180	None
	Std dev:	35.18	0	20	None

Message	Time	Id	Solution	Type	Fror
Autosave complete.	11/18/2022 4:30:45 PM		1robot_2linear_predefi	Info	RS
Inspection_1 is updated.	11/18/2022 4:29:59 PM		1robot_2linear_predefi	Info	HMI
Inspection_1 is updated.	11/18/2022 4:29:55 PM		1robot_2linear_predefi	Info	HMI
Inspection_1 is updated.	11/18/2022 4:28:47 PM		1robot_2linear_predefi	Info	HMI
Inspection_1 is updated.	11/18/2022 4:28:42 PM		1robot_2linear_predefi	Info	HMI

xx2200001128

8 Click **OK**.

To classify the inspection area as accepted or rejected the histogram tool evaluates two different magnitudes within the specified region:

Mean defines the min and max value for the inspection model. If the inspection area has a mean value less than min or higher than max the inspection area will be classified as rejected.

Std dev is a statistical measure that illustrates how closely all the various pixel values are clustered around the mean value. An even color tone gives a narrow histogram with low standard deviation while a speckled pattern gives a wide histogram and a high value for **Std dev**.

Histogram subinspection parameters in item targets

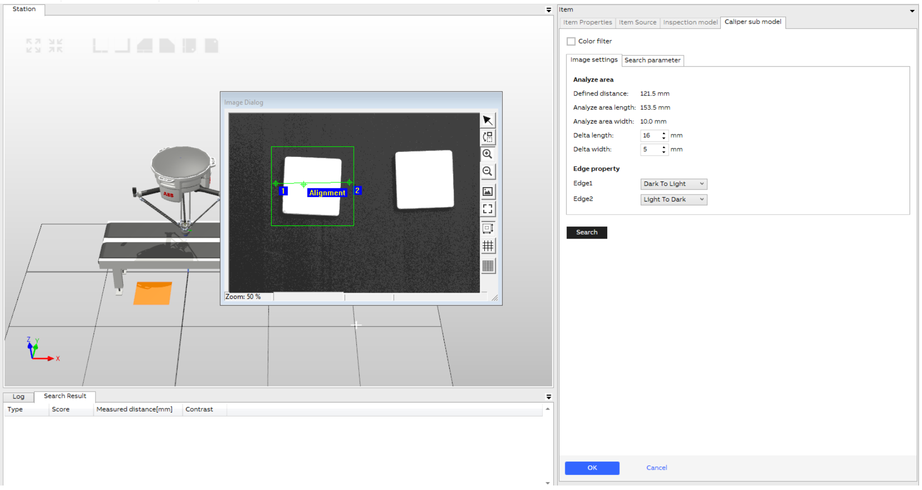
The `Mean` and `standard deviation` parameters can be selected for the target storage.

Continues on next page

Caliper

The *Caliper* tool identifies edges and measures the distance between them. The analysis is only done within the corresponding inspection area. To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. For more information, see *Using color vision on page 82*.

1. Click **+ Sub Inspection Model** to open the **Select Model Type** drop-down list.
2. Select **Caliper** in the drop-down list to create the sub model.
3. Click **OK** on the popped-up dialog to edit area.
4. Drag the rectangle so it covers the pattern.
5. Click **Edit** icon under **Action** to open the Caliper model editing window.
6. Move the line so the end points are located on the edges of the area under the **Image settings**.



□ Color filter

Analyze area

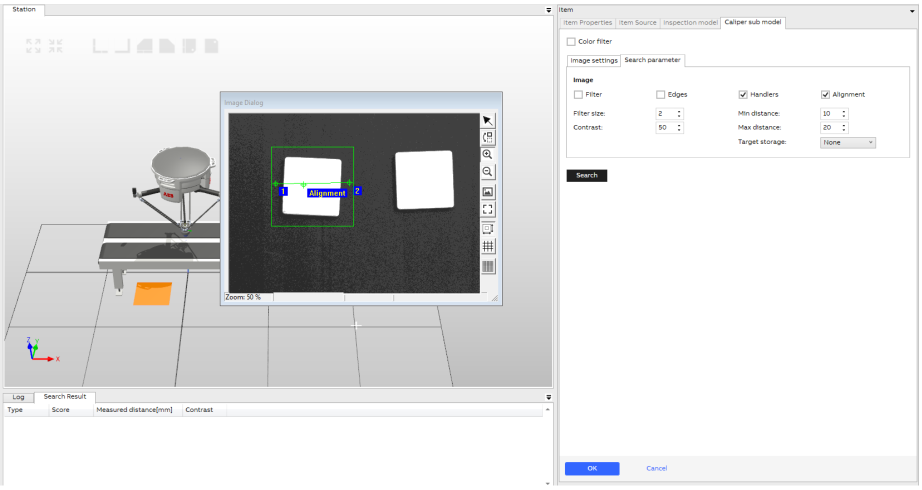
Defined distance: 121.5 mm
Analyze area length: 153.5 mm
Analyze area width: 10.0 mm
Delta length: 16 mm
Delta width: 5 mm

Edge property

Edge1: Dark To Light
Edge2: Light To Dark

xx2200001119

7. Adjust the parameters in the **Search parameter** according to the **Defined distance** in the **Analyze area**.



□ Color filter

Image

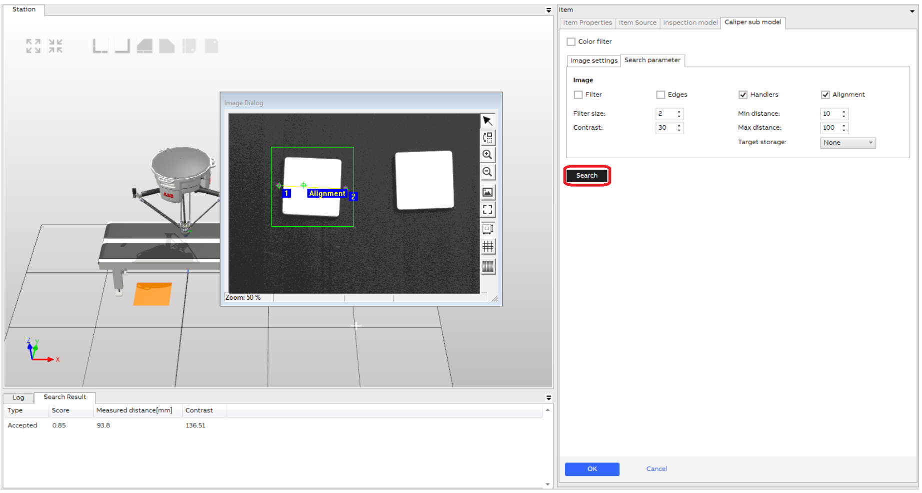
□ Filter □ Edges ☑ Handlers ☑ Alignment

Filter size: 2
Contrast: 50
Min distance: 10
Max distance: 20
Target storage: None

xx2200001120

8. Click **Search**.

Continues on next page



□ Color filter

Image

□ Filter □ Edges ☑ Handlers ☑ Alignment

Filter size: 2

Contrast: 30

Min distance: 10

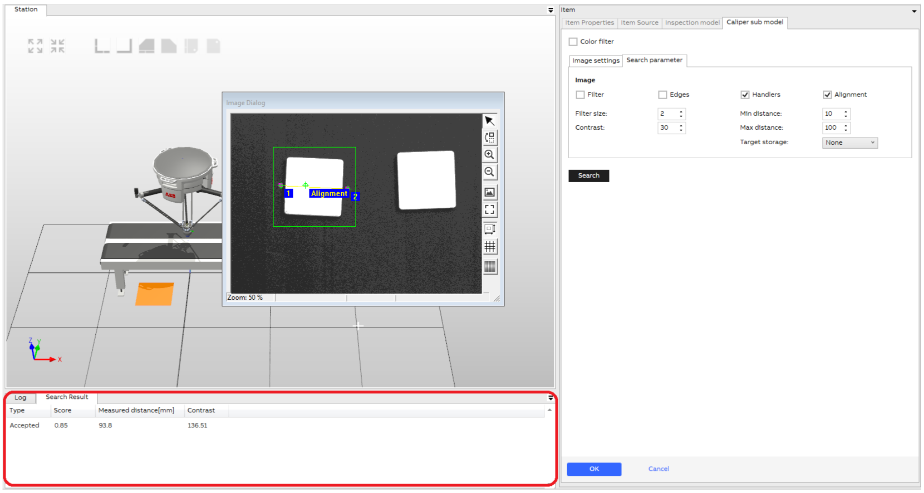
Max distance: 100

Target storage: None

Type	Score	Measured distance[mm]	Contrast
Accepted	0.85	93.8	136.51

xx2200001121

The result is displayed in the **Search Result** tab.



□ Color filter

Image

□ Filter □ Edges ☑ Handlers ☑ Alignment

Filter size: 2

Contrast: 30

Min distance: 10

Max distance: 100

Target storage: None

Type	Score	Measured distance[mm]	Contrast
Accepted	0.85	93.8	136.51

xx2200001122

9 Click **OK**.

To make a *Caliper* analysis a rectangle is defined around the search line.

Defined distance is the distance between the end points of the green line located in the **Image Dialog**. Move the line so the end points are located on the edges of the area.

Analyze area length is the length of the rectangle within which the Caliper analysis will be performed. To increase the **Analyze area length** either increase the **Delta length** value or resize the **Defined distance** line.

Analyze area width is the width of the rectangle within which the Caliper analysis will be performed. To increase the **Analyze area width** increase the **Delta width** value.

Delta length define the extra mm to add to the **Defined distance** to get an **Analyze area length**.

```
Analyze area length=2*Delta length + Defined distance
```

Delta width defines the width of the analyze area.

Continues on next page

```
Analyze area width=2*Delta width
```

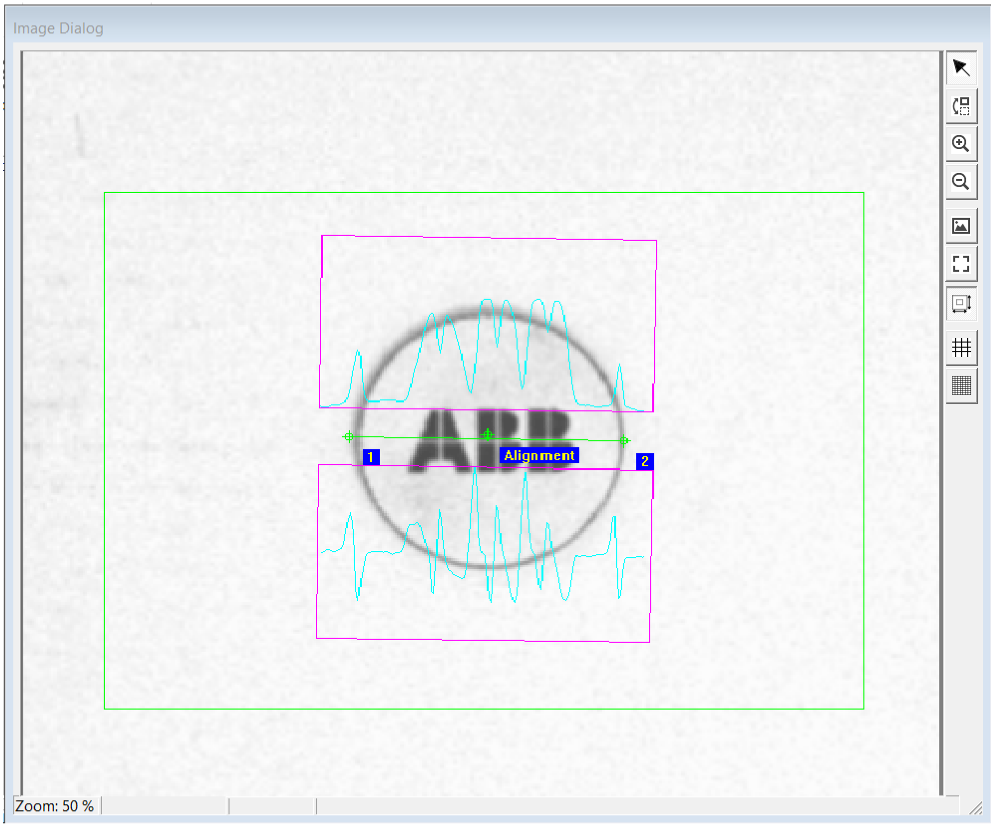
From the analyze area a production image is created. The operation sums all the information in the analyze area, accentuating the strength of edges that lie parallel to the **Analyze area width** and reducing the effects of noise.

Edge property defines the polarity of the edge. The polarity is defined as the measure from **Edge1** to **Edge2**.

The **Search parameter** defines filters using a Gaussian curve. The filter controls how the *Caliper* tool removes noises, how it accentuates the peaks of interest in the image, contrast, and distance.

The **Search** is used to search for two edges with the specified distance (**Defined distance**) and the defined polarity.

The checkboxes in the **Search parameter** define which results should be displayed in the **Image Dialog**.



xx2200001123

Caliper subinspection parameters in term targets

The `Distance` parameter can be selected for the target storage.

External model

This function is reserved for next version.

4.5 How to use color vision

4.5.1 Using color vision

Introduction to color vision

PickMaster Recipe Manager can either be used with monochrome or color cameras. The difference between the two is that an image acquired with a color camera represents each pixel with three 8-bit values (decimal 0-255) instead of only one 8-bit value for monochrome (grayscale) images. In a monochrome image the 8-bit value represents the gray level from white to black, whereas in a color image the three values represent the content of three separate color channels. These three channels represent red, green, and blue (color space RGB) or hue, saturation, and intensity (color space HSI). Which color space to work with, depends on the content of the image.

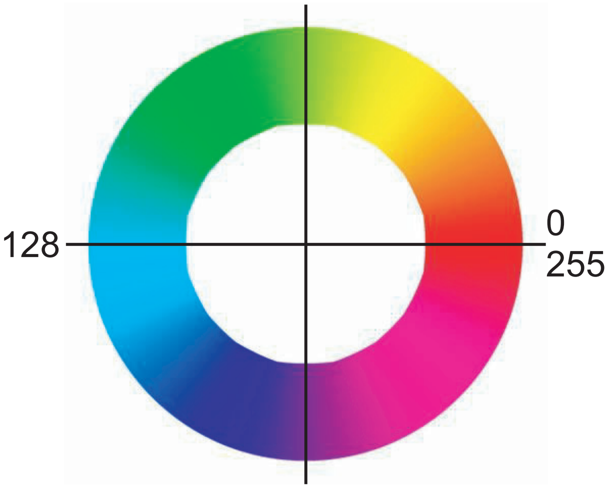
Color spaces

When working with RGB the color of each pixel is represented by its content of red, green, and blue. The numerical representation is straightforward for the three base colors - red (255, 0, 0) green (0, 255, 0), and blue (0, 0, 255). However, it can be difficult to understand the composition of other mixed colors.

HSI is a color space that is more easily translated to the human perception of colors.

- Hue: The location of the color on the on the electromagnetic spectrum. See graphic below.
- Saturation: The purity of the color.
- Intensity: The brightness of the color.

Because the hue spectrum wraps around (both 0 and 255 represent red), it is suitable to display it as a circle.



xx2100002336

When using color filtering it is easier to distinguish between colors if they are dissimilar. The level of similarity may be interpreted as the distance between the colors in color space. The difference may be more pronounced in one or the other

Continues on next page

of the two color spaces and for this reason it is wise to try out filters in both color spaces.

Lighting

Because a color system provides more information about the color contents of an image it is also more sensitive to lighting conditions. It is very important to provide uniform light, that is consistent over time.

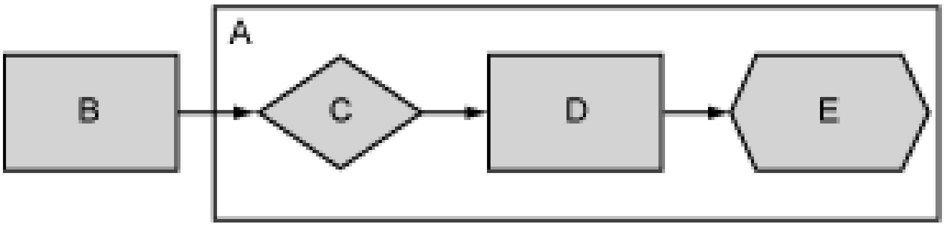
Computer performance

Color vision is very resource consuming: acquisition, warping, and filtering all take more time. It is important to keep the number of cameras and frame rate moderate. The performance limit can vary greatly as it is a combination of the vision task and the computer resources.

Color vision in PickMaster Recipe Manager

PickMaster Recipe Manager provides color vision in the form of a filter. This filter is accessible from the *Geometric*, Blob and Caliper configuration dialogs, both as standalone, alignment and sub-inspection models. The filter is a pre-processing step which takes place before the object recognition or measurement. Every model can have its own individual filter setting.

The camera acquires a color image, that is converted into a grayscale image by passing it through a color filter, as shown in the following figure.

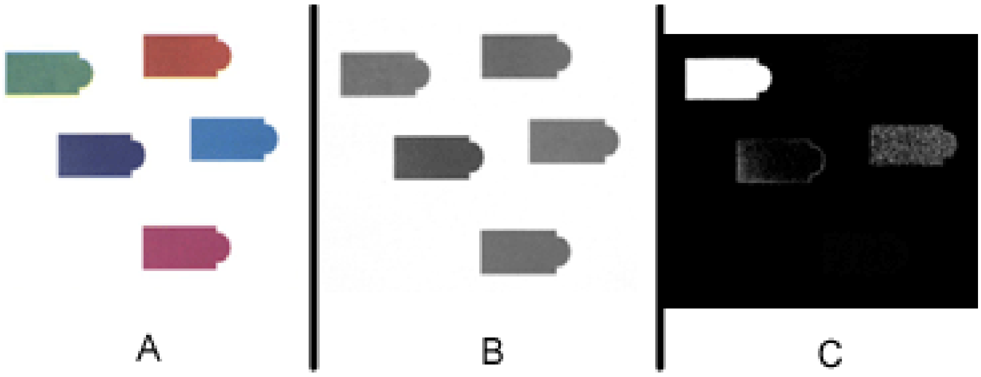


xx0900000445

A	Vision model
B	Color image
C	Color filter
D	grayscale image
E	Object recognition

The result of the color filter is a grayscale image in which certain colors have been accentuated or attenuated according to the filter settings. The object recognition tools (*Blob*/*Geometric*) operate on this grayscale image.

Continues on next page



xx0900000446

A	An image acquired with a color camera.
B	The same scene acquired with a monochrome camera.
C	The color image after having passed through a filter which is set to extract green. This is the image that will be used by *Geometric*/*Blob*.

Prerequisites

The camera must be a color camera.

The color video format must be configured for the camera.

The Cognex vision license must contain the color tool option.

Calibrating the camera's white balance

A camera is delivered with default settings. These include three parameters which represent the white balance of the camera. Depending on the light source, the image can get an undesired color tone. Different light sources emit light of different temperatures (color content) and the camera needs to be color calibrated in order to compensate for this light.

The basic concept is to present the camera with a gray scene, that is a scene that has equal contents of red, green, and blue. The most accurate method is to take a sheet of white paper and adjust the light settings of the camera in order to make the scene appear gray.

Use this procedure to calibrate the white balance for the camera.

1. In the tree view, right-click on the camera and select **Configuration**.

   The **Camera Configuration** dialog is opened
2. Place a white sheet of paper under the camera. The sheet must cover the whole field of view.
3. Adjust the light settings (aperture or exposure time) to make the scene appear mid-gray. The number of saturated pixels (completely black or white) should be kept to a minimum.
4. Press Calculate. This will calculate the white balance calibration parameters.
5. Click **Apply**.

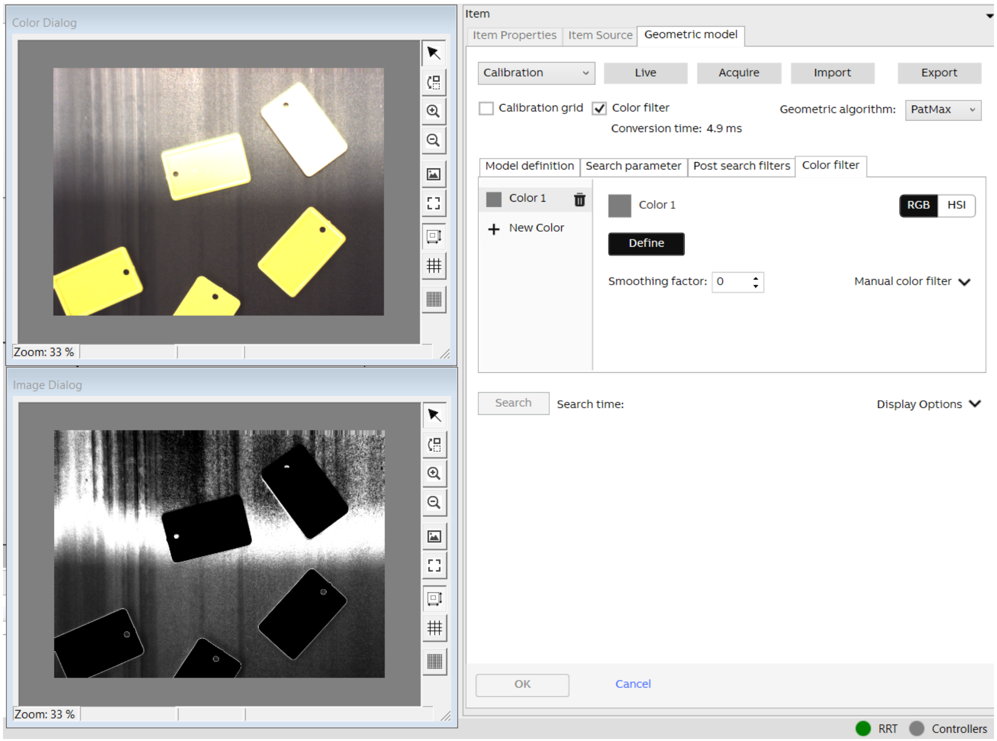
   The camera's internal settings are now modified. If the calibration is successful the color image and the grayscale image of the white paper sheet should now look the same (gray).

Continues on next page

6 Click **OK**.

The settings are stored in the camera. If the parameters are not saved, the camera will loose the calibration when PickMaster Recipe Manager is restarted.

Illustration Color Filter Settings



☐ Calibration grid ☑ Color filter

Conversion time: 4.9 ms

Geometric algorithm: PatMax

Smoothing factor: 0

xx2100002268

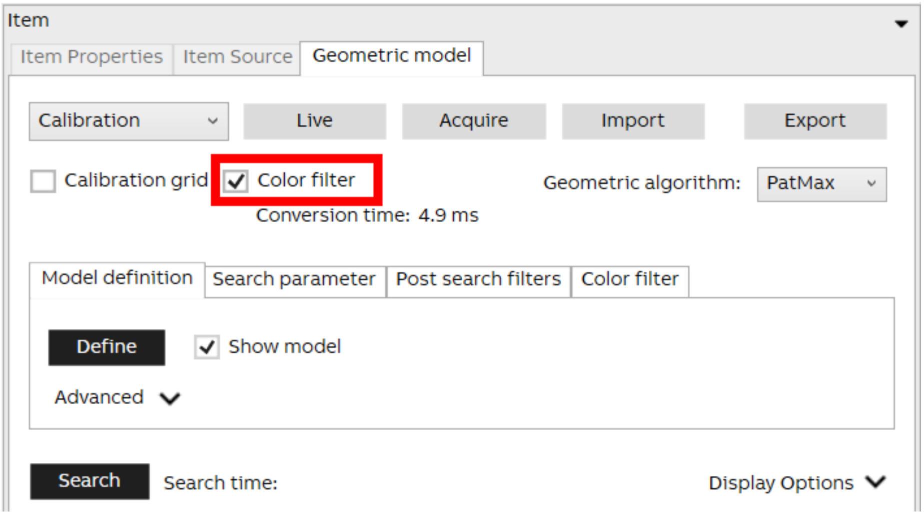
Configuring color vision

The *Geometric* and *Blob* configuration dialogs contain a checkbox to enable color filtering (**Color filter**), and a tab page to display the filter settings.

Use this procedure to configure color vision.

1 In the *Geometric* or *Blob* configuration dialog, select **Color Filter**. This will enable the filter.

Continues on next page



Calibration

□ Calibration grid ☑ Color filter

Conversion time: 4.9 ms

Geometric algorithm: PatMax

☑ Show model

Search time:

xx2100002266

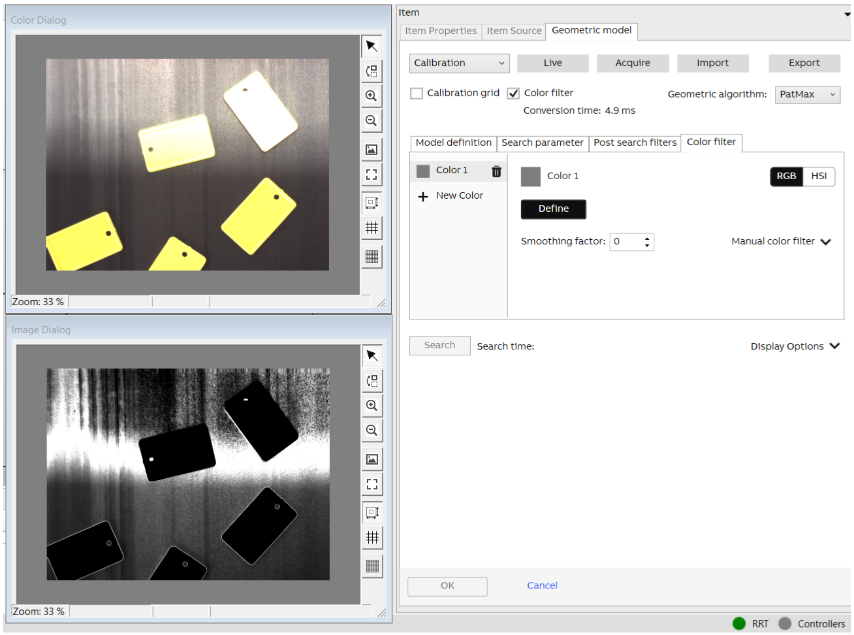
The **Color Filter Settings** tab is opened together with a second video window showing the color image.

2 In the **Color Filter** tab, select **RGB** or **HSI**.

3 In the **Define color** tab, color samples can be collected from the display to indicate which colors should be enhanced.

a Click **Define.** An adjustable rectangle will appear in the color dialog.

b Move/resize the rectangle to indicate what color should pass through the filter. The indicated color range will be converted to white in the output grayscale image. Colors that are dissimilar to the specified color will be converted to black.



Calibration

□ Calibration grid ☑ Color filter

Conversion time: 4.9 ms

Geometric algorithm: PatMax

Color 1

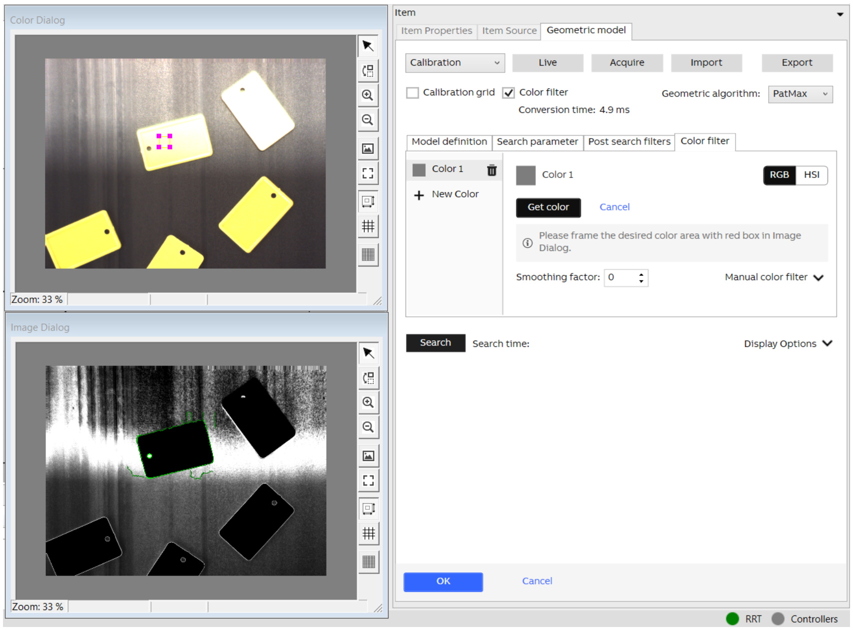
Smoothing factor: 0

Search time:

xx2100002268

c Click **Get color** to store this color range.

Continues on next page



Geometric algorithm: PatMax

Conversion time: 4.9 ms

Please frame the desired color area with red box in Image Dialog.

Smoothing factor: 0

xx2100002269

4 In the **Manual color filter** tab, adjust each color channel to improve the result if needed.

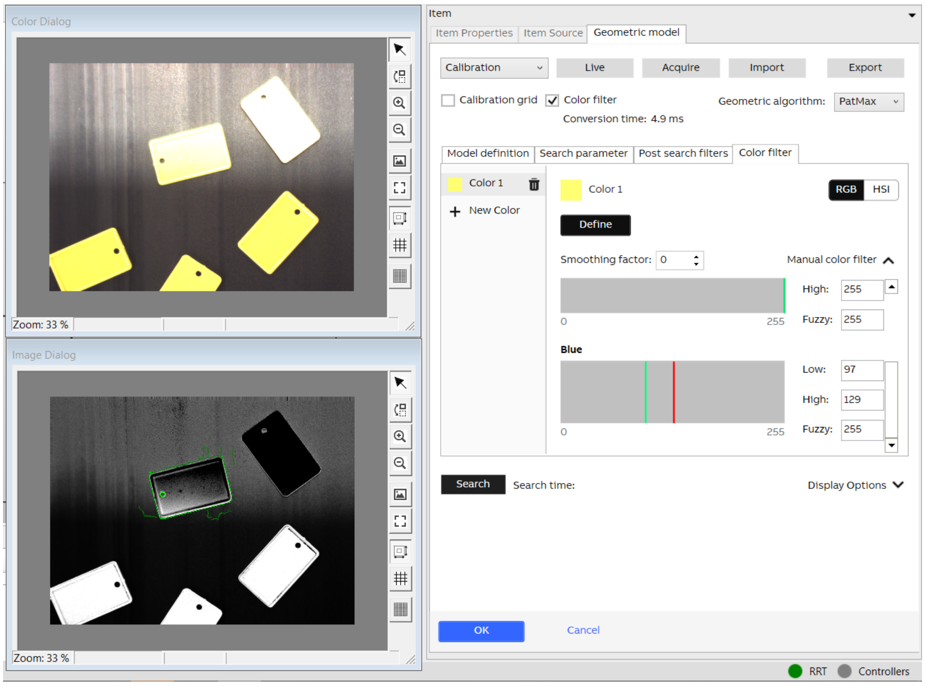
- **Low** specifies the lower limit of the color range that will translate into white pixels in the output image. Minimum is 0 and maximum is 255.
- **High** specifies the upper limit of the color range that will translate into white pixels in the output image. Minimum is 0 and maximum is 255.
- **Fuzzy** specifies how colors outside the minimum and maximum thresholds should be filtered to the output grayscale image. A value of 0 indicates that colors outside the range specified by Low and High will be completely removed by the filter - the result is a black and white image. A non-zero value means that colors outside the Low/High range will be weighted in the output image. A higher value produces a smoother grayscale image. Minimum is 0, maximum is 255.

5 If needed, add a new color range to the list in the **Colors** section.

Each pixel of the output image is computed as the corresponding maximum output pixel of all individual color range filters.

6 If needed, adjust the smoothing factor to reduce noise in the resulting grayscale image.

Continues on next page



□ Calibration grid ☑ Color filter

Geometric algorithm: PatMax

Conversion time: 4.9 ms

Smoothing factor: 0

Manual color filter

High: 255

Fuzzy: 255

Blue

Low: 97

High: 129

Fuzzy: 255

Search time:

xx2100002270

7 Proceed to define the object recognition model.

> **Tip**
>
> Filter ranges should be narrow to provide an output image with high contrast. From an image quality perspective, it is often better to select small homogeneously colored samples and add several ranges to the list of colors.

> **Tip**
>
> Try to filter with both RGB and HSI. Sometimes one may work significantly better than the other.

Example 1

This example describes how to locate a part with *Geometric* and inspect the color with *Blob*.

1 Create an inspection model, see *Configuring inspection models on page 73*.

2 Create a *Geometric* alignment model. Use color filtering if contrast needs to be increased, or use the unfiltered monochrome image if there is sufficient contrast.

3 Add a *Blob* sub inspection model.

   a Select **Color filter** checkbox. This opens the **Color Filter Settings** tab.

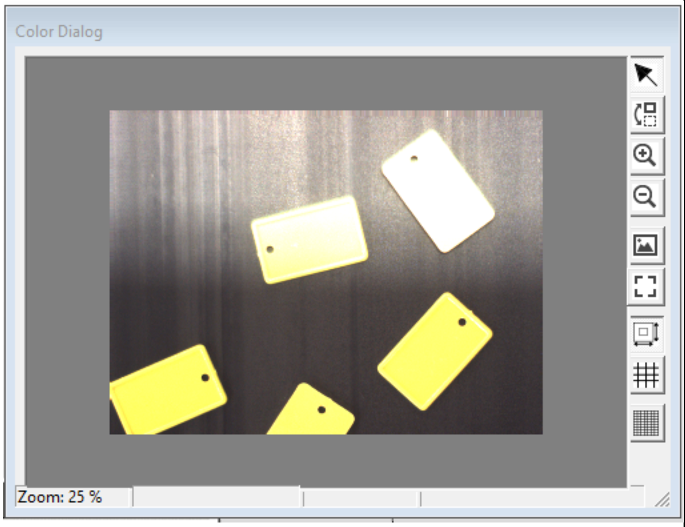
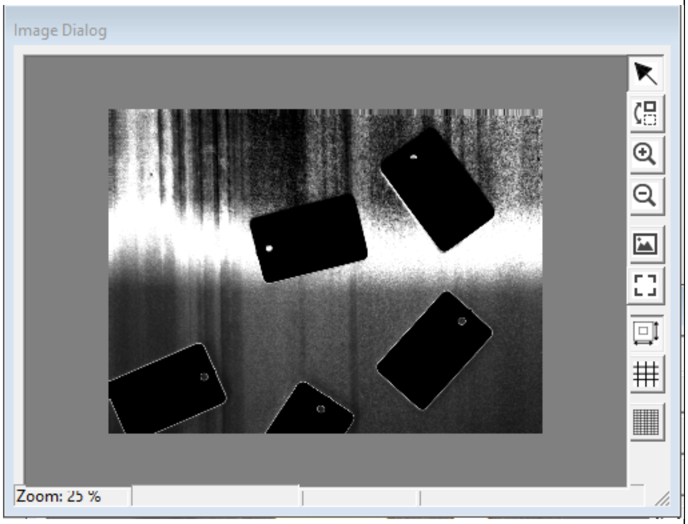
   b Extract the color to be inspected by clicking **Define color**. This filters the desired color into white in the Blob image window.

   c Switch to other tab to do further configuration.

Continues on next page

   d. Adjust the Blob settings so as to find the white blob.
   e. If necessary, adjust the settings of the color filter and the Blob analysis.
4. Test the result in the Inspection Configuration dialog.

Example 2

Color image before filter	 xx1900000946
Black and white image after filter	 xx1900000947

4.6 How to work with user script

4.6.1 User script

Introduction

The **User Script** is a software component provided by PickMaster Twin for users to integrate their custom function.

With this function, user can customize the item position generation, adjustment, filter, or distribution according to their own requirements to achieve user-defined picking and placing of items. For example, the **User Script** can be queried for positions instead of using predefined positions. It is also possible for **User Script** objects to adjust item positions generated by vision models in PickMaster Recipe Manager. Item positions carry some free usage parameters that can be set by the user script. These parameters can later on be accessed in RAPID by the robot that handles the position.

Note

Only qualified personnel should write or modify the script files.

It is the responsibility of the writer to make sure that the cell is safe when running with the script files.

Note

Only Python 3.12 is supported in PickMaster® Twin products.

When installing PickMaster Twin products, Python 3.12 will be installed to your computer and the path of Python 3.12 will be added to the system environment variables automatically.

Tip

Syntax errors will cause the script files fail to run.

With the following way to avoid the syntax errors:

1. Keep to use the same editor for the same script file.
2. It is recommended to use PyCharm or Notepad++ to edit the script files, as they have syntax checking capabilities for Python files.

CAUTION

It is the responsibility of the integrator to implement that local presence is set up in a correct way.

It is the responsibility of the integrator to implement that single point of control is set up in a correct way.

Continues on next page

DANGER

Protect the script carefully if it is used in the production.

Anyone who has access to the script can modify the script directly. This may cause serious danger.

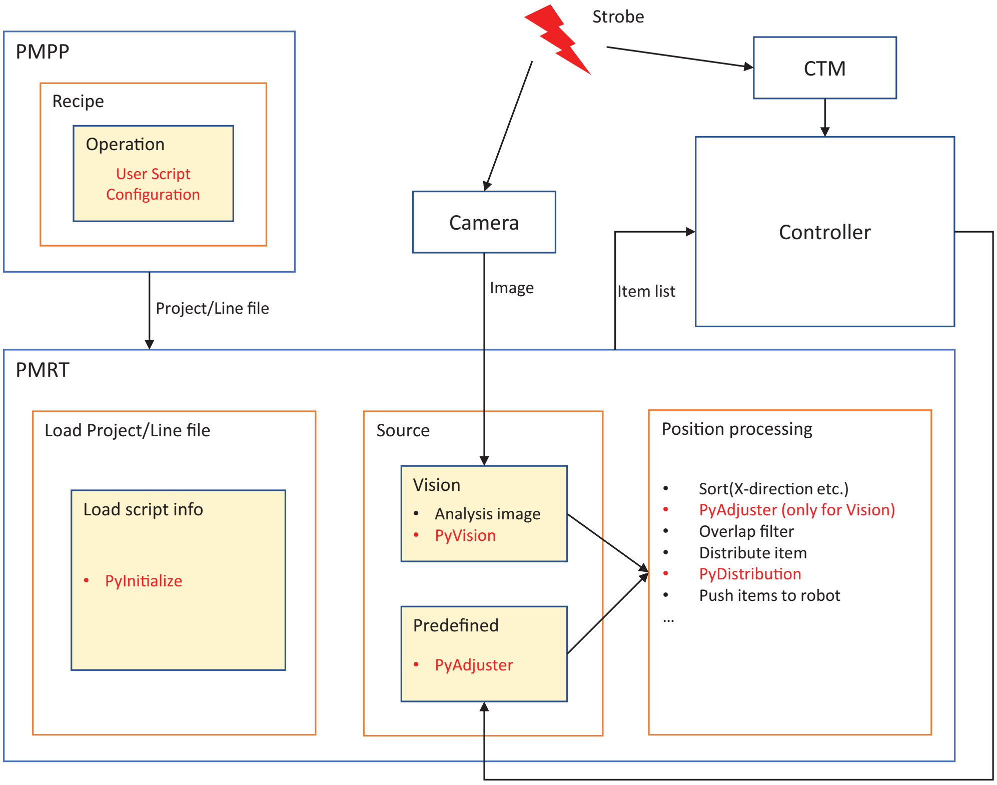
Note

The user script and external sensor cannot be used at the same time in one recipe.

Note

Python script files will not be included in the Pack&Go file. Copy the Python script files to the desired destination.

Flow chart



xx2400000705

Application scenario

User script is an advanced feature provided by PMTW to users, which can be used in the following scenarios:

1. Item generation and deleting

   Users can customize the generation and placement of items in the script according to the requirements to meet the actual needs of customers.

Continues on next page

2 Adjust the picking and placing position of items

Users can adjust the position of items statically or in real time in the script according to requirements, meeting users' requirements for real-time adjustment and high precision of material positions.

3 Items filtering and sorting

The user can filter and sort the current items in the script according to the requirements to meet the user's requirements for item screening and capture sequence.

4 Adjustment of item distribution strategy

The current distribution strategies are LoadBalance and ATC. The user can adjust the current distribution strategy to meet the requirements of user-defined distribution.

5 Item identification

When the vision interface is used, the user can further process the pictures taken by the camera and identify new item information.

6 Bind additional information

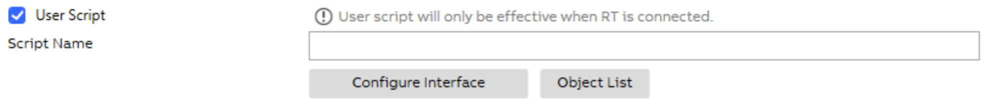
Five optional parameters are provided in the interface parameters. Users can configure optional parameters to bind some additional information with the material and send it to the robot through the software to achieve some special functions, such as item code binding and item tracking.

User value

This function expands the application scenarios of the software. Users can customize the standard functions of the software according to their own needs, which can realize the functions of custom generation, picking, placing, sorting, filtering, and distribution of objects, to meet the needs of users for various specific application scenarios, improve the picking accuracy and production efficiency, and create more value for users.

Configuration overview

When the **User Script** checkbox is selected, the **User Script** setting content will show up.



☑ User Script

Script Name

User script will only be effective when RT is connected.

xx2200001779

	Description
Script Name	Type the predefined script file name with `.py`. **Tip** The predefined script file(s) should be put into `C:\Users\xxxx\Documents\PickMaster\PMScripts` folder before use any script function.

Continues on next page

	Description
Configure Interface	Select which user script interface to be used. Four types user script interfaces are supported by PickMaster Twin.
Object List	Show all available objects (Name and ID) in current operation.

Supported **User Script** interface types overview

PickMaster Twin supports four types of **User Script**.

User script interface	Description
Initialize Interface	This interface is used to provide the user to initialize the User Script program, such as: initialize the parameters, etc. **Tip** **Initialize Interface** will be executed only once when the the **Start** is clicked. The other three interfaces will be executed when DSQC 2000 or DSQC 377 signals are triggered. For more details, see *Initialize Interface PyInitialize: Initialize data on page 95*.
Adjuster Interface	This interface is used to provide the user to realize the customized item position generation and adjustment. Each time the model generates positions, an array with the positions is sent to the **User Script** object. The **User Script** object can then control the positions in any desired way. Positions can be changed, removed, or added. For more details, see *Adjuster Interface PyAdjuster: Modify position on page 96*.
Vision Interface	This interface is used to provide the user to realize the customized item position filter and adjustment by vision result. This interface will be invoked when the Runtime execute to the item recognition section in production. **Tip** The **Vision Interface** can only be used in Production. The other three interfaces can be used in Production and Simulation. For more details, see *Vision Interface PyVision: Recognize items by reanalyzing image on page 97*.
Distribution Interface	This interface is used to provide the user to realize the customized distribution function. This interface will be invoked when the item distribution executes. For more details, see *Distribution Interface PyDistribution: Adjust the target items information after distribution and before push them to robot on page 101*.

Configuring the **User Script** function

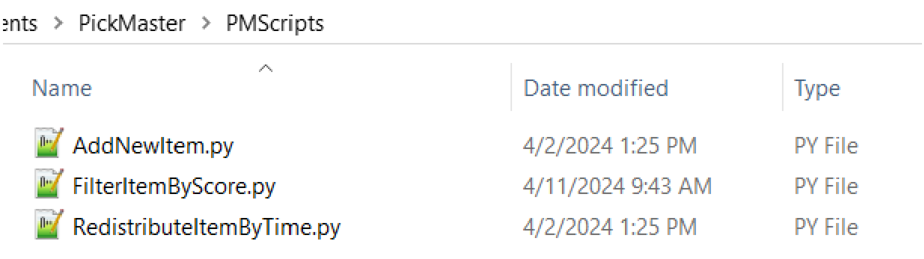
Follow this procedure to configure the user script function:

1 Put the predefined script files into the destination folder.

Continues on next page

Tip

The predefined script file(s) should be put into `C:\Users\xxxx\Documents\PickMaster\PMScripts` folder before use any script function.


ents > PickMaster > PMScripts

Name	Date modified	Type
AddNewItem.py	4/2/2024 1:25 PM	PY File
FilterItemByScore.py	4/11/2024 9:43 AM	PY File
RedistributeItemByTime.py	4/2/2024 1:25 PM	PY File

xx2200001784

2 Select the **User Script** checkbox in PickMaster Powerpac **Recipe** setting to open the configuration page.

3 Input the predefined script file name into the **Script Name** text box.

4 Click **Configure Interface** to open the interface type page.

5 On the popped-up page, select the desired interface type.

Tip

The four types can be used at the same time.

6 Click **Done** to finish the user script function setting in PickMaster Powerpac.

7 Set the time out value in Runtime configuration file *PickMasteru.exe.config*.

For more information about time out setting, see *Timeout setting for user script on page 95*.

Tip

The destination folder of the Runtime configuration file *PickMasteru.exe.config*:

- VRT: `C:\Program Files (x86)\ABB\PickMaster Twin 3\PickMaster Twin Runtime 3\PickMaster VirtualRuntime`
- RRT: `C:\Program Files (x86)\ABB\PickMaster Twin 3\PickMaster Twin Runtime 3\PickMaster Runtime`

Continues on next page

Timeout setting for user script

Set the execution time limit of user scripts to avoid PickMaster Twin product exceptions caused by excessive execution time of the user scripts.

Template	Key	Value (In the template)	Explanation
`<add key="PyInitializeTimeout" value ="1500"/>`	`PyInitializeTimeout`	**x (1500)**	The timeout of the PyInitialize interface is **x (1500)** ms. When the executing time exceeds the set value, an warning will display in the log view.
`<add key="PyAdjusterTimeout" value ="1500"/>`	`PyAdjusterTimeout`	**x (1500)**	The timeout of the PyAdjuster interface is **x (1500)** ms.
`<add key="PyVisionTimeout" value ="1500"/>`	`PyVisionTimeout`	**x (1500)**	The timeout of the PyVision interface is **x (1500)** ms.
`<add key="PyDistributionTimeout" value ="1500"/>`	`PyDistributionTimeout`	**x (1500)**	The timeout of the PyDistribution interface is **x (1500)** ms.
`<add key="MaxTimeoutCount" value ="5"/>`	`MaxTimeoutCount`	**x (5)**	The maximum consecutive timeouts of each interface is **x (5)** times. When the number of consecutive timeouts exceeds the set maximum value, Runtime will stop the interface function calling, clear all objects and display the error log to notify the user to stop the station and check the script.

User script interface

Initialize Interface PyInitialize: Initialize data

This interface is used to initialize the script, and transfer current RT information, item information, container information, and workarea information to the user script, which can be processed by the user, such as creating a new item. At the same time, users can add user program initialization operations in this interface, such as starting external programs, etc., which can be started at the same time when starting the station.

Argument	Description	Explanation	In the example:
type	Runtime type	• 0:VRT • 1:RRT	

itemInfo	Item information which contains `{Key}:{Name:{} Id:{}}`
	For example:
	`itemInfo=`
	`{`
	`'0':{'Name':Item_1, 'Id':'[illegible]'},`
	`};`
		`{Key}`
		Key: unique index
			`'0'`

		`Name:{}` Name: name of the item	`'Name':Item_1`
		`Id:{}` Id: ID of the item	`'Id':'[illegible]'`

For more example, see *`AddNewItem.py` on page 103* and *`RedistributeItemByTime.py` on page 107*.

Continues on next page

Adjuster Interface PyAdjuster: Modify position

Argument	Description	Explanation	In the example:

items	Item information, which contains `Time:{} {Key}:{X:{} Y:{} Z:{} RX:{} RY:{} RZ:{} Tag:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{}}`.
	For example:
	`items =`
	`{`
	`'Time': 1666849507.969,`
	`'0': {'X': 0.0,`
	`'Y': 150.0,`
	`'Z': 0.0,`
	`'RX': 0.0,`
	`'RY': 0.0,`
	`'RZ': 0.0,`
	`'Tag': 0,`
	`'Val1': 0.0,`
	`'Val2': 0.0,`
	`'Val3': 0.0,`
	`'Val4': 0.0,`
	`'Val5': 0.0,`
	`'Level': 2,`
	`'Id': '35139a6c-56a8-437d-b180-7f40e49bf6ff'`
	`}`
	`}`
		`Time:{}`
		Time: time stamp(s), get the number of milliseconds since 1 Jan 1970
			`'Time': 1666849507.969,`

		`{Key}` Key: unique index	`'0'`
		`X:{}` X: the location value of the item in X direction	`'X': 0.0`
		`Y:{}` Y: the location value of the item in Y direction	`'Y': 150.0`
		`Z:{}` Z: the location value of the item in Z direction	`'Z': 0.0`
		`RX:{}` RX: the rotation angle value of the item in X direction	`'RX': 0.0`
		`RY:{}` RY: the rotation angle value of the item in Y direction	`'RY': 0.0`
		`RZ:{}` RZ: the rotation angle value of the item in Z direction	`'RZ': 0.0`
		`Tag:{}` Tag: used in rapid	`'Tag': 0`
		`Val1:{}` Val1, Val2, Val3, Val4, Val5: optional value, used in rapid	`'Val1': 0.0`
		`Val2:{}` Val2: optional value, used in rapid	`'Val2': 0.0`
		`Val3:{}` Val3: optional value, used in rapid	`'Val3': 0.0`
		`Val4:{}` Val4: optional value, used in rapid	`'Val4': 0.0`
		`Val5:{}` Val5: optional value, used in rapid	`'Val5': 0.0`
		`Level:{}` Level: inspection level	`'Level': 2`

Continues on next page

Argument	Description	Explanation	In the example:
		- 0: Discarded - 1: Rejected - 2: Accepted	

		`Id:{}`
		Id: ID of the item
			`'Id':`
			`'35139a6c-56a8-437d-b180-7f40e49bf6ff'`
			`}`

For more example, see `AddNewItem.py` *on page 103*.

Vision Interface PyVision: Recognize items by reanalyzing image

Argument	Description	Explanation	

imageData	Image data, which contains `Width:{}` `Height:{}` `IsColor:{}` `Grey:{}` `Blue:{}` `Green:{}` `Red:{}`
	For example:
	`Grey image`
	`imageData =`
	`{`
	`'Width': 481,`
	`'Height’: 409,`
	`'IsColor’: 0,`
	`‘Grey’: [56,…,67]`
	`}`
	`Colorful image`
	`imageData =`
	`{`
	`'Width': 481,`
	`'Height’: 409,`
	`'IsColor’: 1,`
	`‘Blue’: [56,…,67],`
	`‘Green': [56,…,67],`
	`‘Red': [56,…,67]`
	`}`
		`Width:{}`
		- Width: image width in pixel
			`'Width': 481,`

		`Height:{}` - Height: image height in pixel	`'Height’: 409,`

		`IsColor:{}`
		- `Grey:{}`
		- `Blue:{}` `Green:{}` `Red:{}`.
		- IsColor:
		  - 0: Grey image
		  - 1: Colorful image
		- Grey: grey data, valid from 0 to 255
		- Blue: blue data, valid from 0 to 255
		- Green: green data, valid from 0 to 255
		- Red: red data, valid from 0 to 255
			For `Grey image`
			`'IsColor’: 0,`
			`‘Grey’: [56,…,67]`
			For `Colorful image`
			`'IsColor’: 1,`
			`‘Blue’: [56,…,67],`
			`‘Green': [56,…,67],`
			`‘Red': [56,…,67]`

Continues on next page

Argument	Description	Explanation	

calibData	Calibration data, which contains `UpperLeftX:{}` `UpperLeftY:{}` `LowerRightX:{}` `LowerRightY:{}` `XScale:{} YScale:{}`.
	For example:
	`calibData =`
	`{`
	`'UpperLeftX’: -313,`
	`'UpperLeftY’: -265,`
	`'LowerRightX’: 168,`
	`‘LowerRightY’: 144,`
	`‘XScale’: 0.415,`
	`‘YScale': 0.415`
	`}`
		`UpperLeftX:{}`
		• **UpperLeftX: the upper left point on the X direction in the coordinate system in pixel**
			`'UpperLeftX’: -313,`

		`UpperLeftY:{}` • **UpperLeftY: the upper left point on the Y direction in the coordinate system in pixel**	`'UpperLeftY’: -265,`
		`LowerRightX:{}` • **LowerRightX: the lower right point on the X direction in the coordinate system in pixel**	`'LowerRightX’: 168,`
		`LowerRightY:{}` • **LowerRightY: the lower right point on the Y direction in the coordinate system in pixel**	`‘LowerRightY’: 144,`
		`XScale:{}` • **XScale: X axial scale of real item and image in pixel**	`‘XScale’: 0.415,`
		`YScale:{}` • **YScale: Y axial scale of real item and image in pixel.**	`‘YScale': 0.415`

Continues on next page

Argument	Description	Explanation	
items	Item information, which contains: `Time:{}` and • Geomatric: `{Key}:{X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{} XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{} ModelType:{} Score:{} XScale:{} YScale:{} Contrast:{} FitError:{} Coverage:{} Clutter:{}}` • Blob: `{Key}:{X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{} XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{} ModelType:{} Area:{} Perimeter:{} Elongation:{} Circularity:{}}` • Inspection: `{Key}:{X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{} XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{} ModelType:{}}` For example: Geomatric	`Time:{}` • Time: time stamp(s), get the number of milliseconds since 1 Jan 1970	`'Time': 1666849507.969,`
		`{Key}` Key: unique index	`'0’`
		`X:{}` • X: the location value of the item in X direction	`'X’: -80.1,`
		`Y:{}` • Y: the location value of the item in Y direction	`'Y’: -77.2,`
		`Z:{}` • Z: the location value of the item in Z direction	`'Z': 0.0,`
		`RZ:{}` • RZ: the rotation angle value of the item in Z direction	`‘RZ’: -7.22,`
		`SortValue:{}` • SortValue: sort value	`‘SortValue’: 0.976,`
		`ZValid:{}` • ZValid: - 1: valid - 0: invalid	`‘ZValid': 0,`
		`XImgPos:{}` • XImgPos: item position in image on X direction •	`‘XImgPos': -80.1,`
		`YImgPos:{}` • YImgPos: item position in image on Y direction	`‘YImgPos': -77.2,`

		`Val1:{} Val2:{} Val3:{} Val4:{} Val5:{}`
		• Val1, Val2, Val3, Val4, Val5: optional value, used in rapid
			`'Val1': 0.0,`
			`'Val2': 0.0,`
			`'Val3': 0.0,`
			`'Val4': 0.0,`
			`'Val5': 0.0,`

		`Level:{}` • Level: inspection level - 0: Discarded - 1: Rejected - 2: Accepted	`'Level':2,`

Continues on next page

Argument	Description	Explanation	

	resResult = {
	'Time':
	1666849507.969,
	'0’:{'X’: -80.1,
	'Y’: -77.2,
	'Z': 0.0,
	‘RZ’: -7.22,
	‘SortValue’: 0.976,
	‘ZValid': 0,
	‘XImgPos': -80.1,
	‘YImgPos': -77.2,
	'Val1': 0.0,
	'Val2': 0.0,
	'Val3': 0.0,
	'Val4': 0.0,
	'Val5': 0.0,
	'Level':2,
	'Id':
	'35139a6c-56a8-437d-b180-7f40e49bf6ff’,
	'ModelType':1,
	'Score':0.747174859046936,
	'XScale':0.9995959997177124,
	'YScale':0.9995959997177124,
	'Contrast':12.289325714111328,
	'FitError':0.3699681460857391 4,
	'Coverage':0.747174859046936,
	'Clutter':0.10466811060905457
	}
	}
	Blob
	resResult = {
	'Time':
	1666849507.969,
	'0’:{'X’: -80.1,
	'Y’: -77.2,
	'Z': 0.0,
	‘RZ’: -7.22,
	‘SortValue’: 0.976,
	‘ZValid': 0,
	‘XImgPos': -80.1,
	‘YImgPos': -77.2,
	'Val1': 0.0,
	'Val2': 0.0,
	'Val3': 0.0,
	'Val4': 0.0,
	'Val5': 0.0,
	'Level':2,
	'Id':
	'35139a6c-56a8-437d-b180-7f40e49bf6ff’,
	'ModelType':2,
		Id:{}
		• Id: ID of the item
			'Id':
			'35139a6c-56a8-437d-b180-7f40e49bf6ff’,

		ModelType:{} • ModelType: - 1: Geomatric - 2: Blob - 3: Inspection	**For Geomatric:** 'ModelType':1, **For Blob:** 'ModelType':2, **For Inspection:** 'ModelType':3 }
		For more information, see *Configuring a geometric model with PatMax*. Score:{} • Score: how closely the found item matches the trained model.	'Score':0.747174859046936,
		XScale:{} • XScale: X axial scale of real item and image in pixel	'XScale':0.9995959997177124,
		YScale:{} • YScale: Y axial scale of real item and image in pixel.	'YScale':0.9995959997177124,
		Contrast:{} • Contrast: the image contrast of each item that is found in the image.	'Contrast':12.289325714111328,
		FitError:{} • FitError: a measure of the variance between the shape of the trained pattern and the shape of the pattern found in the search image.	'FitError':0.3699681460857391 4,
		Coverage:{} • Coverage: a measure of the extent to which all parts of the trained pattern are also present in the search image.	'Coverage':0.747174859046936,
		Clutter:{} • Clutter: a measure of the extent to which the found pattern contains features that are not present in the trained pattern.	'Clutter':0.10466811060905457 }
			'Area':0,

Continues on next page

Argument	Description	Explanation	

	`'Area':0,`
	`'Perimeter':0,`
	`'Elongation':0,`
	`'Circularity':0 }`
	`}`
	Inspection
	`resResult = {`
	`'Time': 1666849507.969,`
	`'0’:{'X’: -80.1,`
	`'Y’: -77.2,`
	`'Z': 0.0,`
	`‘RZ’: -7.22,`
	`‘SortValue’: 0.976,`
	`‘ZValid': 0,`
	`‘XImgPos': -80.1,`
	`‘YImgPos': -77.2,`
	`'Val1': 0.0,`
	`'Val2': 0.0,`
	`'Val3': 0.0,`
	`'Val4': 0.0,`
	`'Val5': 0.0,`
	`'Level':2,`
	`'Id': '35139a6c-56a8-437d-b180-7f40e49bf6ff’,`
	`'ModelType':3 }`
	`}`
		For more information, see *Configuring a blob vision model*.
		`Area:{}`
		• Area: expressed in mm^2	

		`Perimeter:{}` • Perimeter: expressed in mm	`'Perimeter':0,`
		`Elongation:{}` • Elongation: the ratio of the feature’s second moment of inertia about its second principal axis to the feature’s second moment of inertia about its first principal axis.	`'Elongation':0,`
		`Circularity:{}` • Circularity: defines the circularity. A value of 1 means perfectly circular and completely filled (no holes).	`'Circularity':0 }`

For more example, see *`FilterItemByScore.py` on page 105*.

Distribution Interface PyDistribution: Adjust the target items information after distribution and before push them to robot

Argument	Description		Explain
WaId	Workarea ID, which contains `WaId:{}`. For example: `WaId = (9B36B3CC-2B65-4954-92A5-3C36D64B5667)`	`WaId:{}` • WaId: Workarea Id	`WaId = (9B36B3CC-2B65-4954-92A5-3C36D64B5667)`

Continues on next page

Argument	Description		Explain

items	Item information, which contains `Time:{} {Key}: {X:{} Y:{} Z:{} q1:{} q2:{} q3:{} q4:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Type:{} Tag:{} Index:{} State:{} Container:{} Layer:{} Group:{} Id:{}}.`
	For example:
	`items =`
	`{`
	`'Time': 1666849507.969,`
	`'0’:{'X': 0.0,`
	`'Y': 150.0,`
	`'Z': 0.0,`
	`‘q1': 0.0,`
	`‘q2’: 1.0,`
	`‘q3': 0.0,`
	`‘q4': 0.0,`
	`'Val1': 0.0,`
	`'Val2': 0.0,`
	`'Val3': 0.0,`
	`'Val4': 0.0,`
	`'Val5': 0.0,`
	`‘Type': 2,`
	`'Tag': 0,`
	`‘Index': 2,`
	`‘State': 0,`
	`‘Container’: 1,`
	`‘Layer’: 1,`
	`‘Group': 0,`
	`'Id':`
	'35139a6c-56a8-437d-b180-7f40e49bdf6ff’
	`}`
	`}`
		`Time:{}`
		• Time: time stamp(s), get the number of milliseconds since 1 Jan 1970
			`'Time': 1666849507.969,`

		`{Key}` Key: unique index	`'0’`
		`X:{}` • X: the location value of the item in X direction	`{'X': 0.0,`
		`Y:{}` • Y: the location value of the item in Y direction	`'Y': 150.0,`
		`Z:{}` • Z: the location value of the item in Z direction	`'Z': 0.0,`
		`q1:{}` • q1, q2, q3, q4: the quaternion values of the item	`‘q1': 0.0,`
		`q2:{}`	`‘q2’: 1.0,`
		`q3:{}`	`‘q3': 0.0,`
		`q4:{}`	`‘q4': 0.0,`

		`Val1:{} Val2:{} Val3:{} Val4:{} Val5:{}`
		• Val1, Val2, Val3, Val4, Val5: optional value, used in rapid
			`'Val1': 0.0,`
			`'Val2': 0.0,`
			`'Val3': 0.0,`
			`'Val4': 0.0,`
			`'Val5': 0.0,`

		`Type:{}` • Index: Index number of the **Accepted Type** or **Rejected Type**	`‘Type': 2,`
		`Tag:{}` • Tag: Used in rapid	`'Tag': 0,`
		`Index:{}` • Index: The sequence number of the current item, which increases with the generation of the item on the conveyor and predefined layout in the container.	`‘Index': 2,`
		`State:{}`	`‘State': 0,`

Continues on next page

Argument	Description		Explain
		- State: item state - 0: Use - 1: Bypass - 2: Used	
		`Container:{}` - Container: container number - 0: it is an item - 1-n: it is a container	`‘Container’: 1,`
		`Layer:{}` - Layer: layer number - 0: it is an item - 1-n: it is layer in the container	`‘Layer’: 1,`
		`Group:{}` - Group: sorting method - 0: None or movement direction - 1: Strict	`‘Group': 0,`

		`Id:{}`
		- Id: ID of the item
			`'Id':`
			'35139a6c-56a8-437d-b180-7f40e49bf6ff’
			`}`

For more example, see *`RedistributeItemByTime.py` on page 107*.

Template

All the user script templates are also provided in the folder `C:\Program Files (x86)\ABB\PickMaster Twin 3\Samples\UserScriptTemplates` when PickMaster Client is installed.

`AddNewItem.py`

```
# PMTW user script demo -- AddNewItem
# Add a new item in the default item list.

# Global definition
RTType = 1
item_1 = r''
container_1 = r''
newObject = {'X': 100.0, 'Y': 50.0, 'Z': 5.0, 'RX': 0.0, 'RY': 0.0,
      'RZ': 0.0, 'Tag': -1, 'Val1': 0.0, 'Val2': 0.0, 'Val3': 0.0,
      'Val4': 0.0, 'Val5': 0.0, 'Level': 2, 'Id':
      '5FAD0398-74F8-4786-BF2A-1225924A8A41'}
# This path need to be created by users.
logPath = r'C:\PMScriptsLog\PickInfo.txt'

# PyInitialize interface
```

Continues on next page

```
def PyInitialize(type, itemInfo):
  global RTType
  global item_1
  global container_1
  RTType = type
  f = open(logPath,'a')
  f.write("PyInitialize\n")
  # RT type
  strLine = "RTType:{}\n".format(str(RTType))
  f.write(strLine)
  # Item information
  keys = itemInfo.keys();
  for key in keys:
    strLine = "{} Name:{} Id:{}\n".format(str(key),
         str(itemInfo[key]['Name']), str(itemInfo[key]['Id']))
    if itemInfo[key]['Name'] == 'Item_1':
      item_1 = itemInfo[key]['Id']
    elif itemInfo[key]['Name'] == 'Container_1':
      container_1 = itemInfo[key]['Id']
    f.write(strLine)
  f.close()
# PyAdjuster interface
def PyAdjuster(items):
  global RTType
  global item_1
  global container_1
  global newObject
  f = open(logPath, 'a')
  f.write("PyAdjuster\n")
  # Modify Id
  newObject['Id'] = item_1
  #newObject['Id'] = container_1

  # Add new item
  iSize = len(items)
  newKey = str(iSize - 1)
  items[newKey] = newObject

  # Item information
  keys = items.keys()
  for key in keys:
    if key == 'Time':
      # Time stamp(s), get the number of milliseconds since 1 Jan
           1970.
      strLine = "Time:{}\n".format(str(items[key]))
      f.write(strLine)
    else:
      # Print
      strLine = "{} X:{} Y:{} Z:{} RX:{} RY:{} RZ:{} Tag:{} Val1:{}
           Val2:{} Val3:{} Val4:{} Val5:{} Level:{}
           Id:{}\n".format(str(key), str(items[key]['X']),
```

Continues on next page

```
           str(items[key]['Y']), str(items[key]['Z']),
           str(items[key]['RX']), str(items[key]['RY']),
           str(items[key]['RZ']), str(items[key]['Tag']),
           str(items[key]['Val1']), str(items[key]['Val2']),
           str(items[key]['Val3']), str(items[key]['Val4']),
           str(items[key]['Val5']), str(items[key]['Level']),
           items[key]['Id'])
      f.write(strLine)
  f.close()
  return items;
```

FilterItemByScore.py

```
# PMTW user script demo -- FilterItemByScore
# Filter item according to score value.

# Global definition
# This path need to be created by users.
logPath = r'C:\PMScriptsLog\PlaceInfo.txt'

# PyVision interface
def PyVision(imageData,calibData,items):
  f = open(logPath, 'a')
  f.write("PyVision\n")
  # Image data
  f.write("ImageData:\n")
  strLine = "Width:{} Height:{}
       IsColor:{}\n".format(str(imageData['Width']),
       str(imageData['Height']), str(imageData['IsColor']))
  f.write(strLine)
  if imageData['IsColor'] == 0:
    strLine = "Grey:{}\n".format(str(imageData['Grey']))
  else:
    strLine =
         "Blue:{}\nGreen:{}\nRed:{}\n".format(str(imageData['Blue']),
         str(imageData['Green']), str(imageData['Red']))
  f.write(strLine)
  # Calibration data
  f.write("CalibrationData:\n")
  strLine = "UpperLeftX:{} UpperLeftY:{} LowerRightX:{}
       LowerRightY:{} XScale:{}
       YScale:{}\n".format(str(calibData['UpperLeftX']),
       str(calibData['UpperLeftY']), str(calibData['LowerRightX']),
       str(calibData['LowerRightY']), str(calibData['XScale']),
       str(calibData['YScale']))
  f.write(strLine)
  # Item information
  f.write("Items:\n")
  keys = items.keys();
  for key in keys:
    if key == 'Time':
      # Time stamp(s), get the number of milliseconds since 1 Jan
           1970.
```

Continues on next page

```
    strLine = "Time:{}\n".format(str(items[key]))
    f.write(strLine)
  else:
    if items[key]['ModelType'] == 1:
      # Geomatric
      strLine = "{} X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{}
           XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{}
           Val5:{} Level:{} Id:{} ModelType:{} Score:{} XScale:{}
           YScale:{} Contrast:{} FitError:{} Coverage:{}
           Clutter:{}\n".format(str(key), str(items[key]['X']),
           str(items[key]['Y']), str(items[key]['Z']),
           str(items[key]['RZ']), str(items[key]['SortValue']),
           str(items[key]['ZValid']), str(items[key]['XImgPos']),
           str(items[key]['YImgPos']), str(items[key]['Val1']),
           str(items[key]['Val2']), str(items[key]['Val3']),
           str(items[key]['Val4']), str(items[key]['Val5']),
           str(items[key]['Level']), items[key]['Id'],
           str(items[key]['ModelType']),
           str(items[key]['Score']), str(items[key]['XScale']),
           str(items[key]['YScale']),
           str(items[key]['Contrast']),
           str(items[key]['FitError']),
           str(items[key]['Coverage']),
           str(items[key]['Clutter']))
      # Filter
      if items[key]['Score'] < 0.8:
        items[key]['Level'] = 0
    elif items[key]['ModelType'] == 2:
      # Blob
      strLine = "{} X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{}
           XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{}
           Val5:{} Level:{} Id:{} ModelType:{} Area:{}
           Perimeter:{} Elongation:{}
           Circularity:{}\n".format(str(key),
           str(items[key]['X']), str(items[key]['Y']),
           str(items[key]['Z']), str(items[key]['RZ']),
           str(items[key]['SortValue']),
           str(items[key]['ZValid']), str(items[key]['XImgPos']),
           str(items[key]['YImgPos']), str(items[key]['Val1']),
           str(items[key]['Val2']), str(items[key]['Val3']),
           str(items[key]['Val4']), str(items[key]['Val5']),
           str(items[key]['Level']), items[key]['Id'],
           str(items[key]['ModelType']), str(items[key]['Area']),
           str(items[key]['Perimeter']),
           str(items[key]['Elongation']),
           str(items[key]['Circularity']))
      # Filter
      if items[key]['Score'] < 0.8:
        items[key]['Level'] = 0
    else:
      # Inspection
      strLine = "{} X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{}
           XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{}
           Val5:{} Level:{} Id:{}
           ModelType:{}\n".format(str(key), str(items[key]['X']),
```

Continues on next page

```
             str(items[key]['Y']), str(items[key]['Z']),
             str(items[key]['RZ']), str(items[key]['SortValue']),
             str(items[key]['ZValid']), str(items[key]['XImgPos']),
             str(items[key]['YImgPos']), str(items[key]['Val1']),
             str(items[key]['Val2']), str(items[key]['Val3']),
             str(items[key]['Val4']), str(items[key]['Val5']),
             str(items[key]['Level']), items[key]['Id'],
             str(items[key]['ModelType']))
      f.write(strLine)
  f.close()
  return items;
```

RedistributeItemByTime.py

```
# PMTW user script demo -- RedistributeItemByTime
# Every minute a robot is exchanged to pick and place items.

import time

# Global definition
RTType = 1
item_1 = r''
item_2 = r''
workarea_2 = r''
workarea_4 = r''
# This path need to be created by users.
logPath = r'C:\PMScriptsLog\PlaceInfo.txt'

# PyInitialize interface
def PyInitialize(type, itemInfo):
  global RTType
  global item_1
  global item_2
  global workarea_2
  global workarea_4
  RTType = type
  f = open(logPath,'a')
  f.write("PyInitialize\n")
  # RT type
  strLine = "RTType:{}\n".format(str(RTType))
  f.write(strLine)
  # Item information
  keys = itemInfo.keys()
  for key in keys:
    strLine = "{} Name:{} Id:{}\n".format(str(key),
         str(itemInfo[key]['Name']), str(itemInfo[key]['Id']))
    if itemInfo[key]['Name'] == 'Item_1':
      item_1 = itemInfo[key]['Id']
    elif itemInfo[key]['Name'] == 'Item_2':
      item_2 = itemInfo[key]['Id']
    elif itemInfo[key]['Name'] == 'ConveyorWorkArea_2':
      workarea_2 = itemInfo[key]['Id']
```

Continues on next page

```
    elif itemInfo[key]['Name'] == 'ConveyorWorkArea_4':
      workarea_4 = itemInfo[key]['Id']
    f.write(strLine)
  f.close()
# PyDistribution interface
def PyDistribution(WaId, items):
  global RTType
  global item_1
  global item_2
  global workarea_2
  global workarea_4
  f = open(logPath, 'a')
  f.write("PyDistribution\n")
  # Workarea Id
  strLine = "WaId:{}\n".format(WaId)
  f.write(strLine)
  # Item information
  keys = items.keys()
  for key in keys:
    if key == 'Time':
      # Time stamp(s), get the number of milliseconds since 1 Jan
           1970.
      strLine = "Time:{}\n".format(str(items[key]))
      f.write(strLine)
    else:
      # Modify before
      strLine = "{} X:{} Y:{} Z:{} q1:{} q2:{} q3:{} q4:{} Tag:{}
           Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Type:{} Index:{}
           State:{} Container:{} Layer:{} Group:{}
           Id:{}\n".format(str(key), str(items[key]['X']),
           str(items[key]['Y']), str(items[key]['Z']),
           str(items[key]['q1']), str(items[key]['q2']),
           str(items[key]['q3']), str(items[key]['q4']),
           str(items[key]['Tag']), str(items[key]['Val1']),
           str(items[key]['Val2']), str(items[key]['Val3']),
           str(items[key]['Val4']), str(items[key]['Val5']),
           str(items[key]['Type']), str(items[key]['Index']),
           str(items[key]['State']), str(items[key]['Container']),
           str(items[key]['Layer']), str(items[key]['Group']),
           items[key]['Id'])
      f.write(strLine)
      # Modify
      if (divmod(time.localtime().tm_min, 2))[1] == 0:
        if WaId == workarea_2:
          items[key]['State'] = 1
        elif WaId == workarea_4:
          items[key]['State'] = 0
      else:
        if WaId == workarea_2:
          items[key]['State'] = 0
        elif WaId == workarea_4:
          items[key]['State'] = 1
```

Continues on next page

```
        # Modify after
        strLine = "{} X:{} Y:{} Z:{} q1:{} q2:{} q3:{} q4:{} Tag:{}
             Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Type:{} Index:{}
             State:{} Container:{} Layer:{} Group:{}
             Id:{}\n".format(str(key), str(items[key]['X']),
             str(items[key]['Y']), str(items[key]['Z']),
             str(items[key]['q1']), str(items[key]['q2']),
             str(items[key]['q3']), str(items[key]['q4']),
             str(items[key]['Tag']), str(items[key]['Val1']),
             str(items[key]['Val2']), str(items[key]['Val3']),
             str(items[key]['Val4']), str(items[key]['Val5']),
             str(items[key]['Type']), str(items[key]['Index']),
             str(items[key]['State']), str(items[key]['Container']),
             str(items[key]['Layer']), str(items[key]['Group']),
             items[key]['Id'])
        f.write(strLine)
    f.close()
    return items;
```

4.7 How to work with products of varying height (2.5D vision)

4.7.1 Working with products of varying height (2.5D vision)

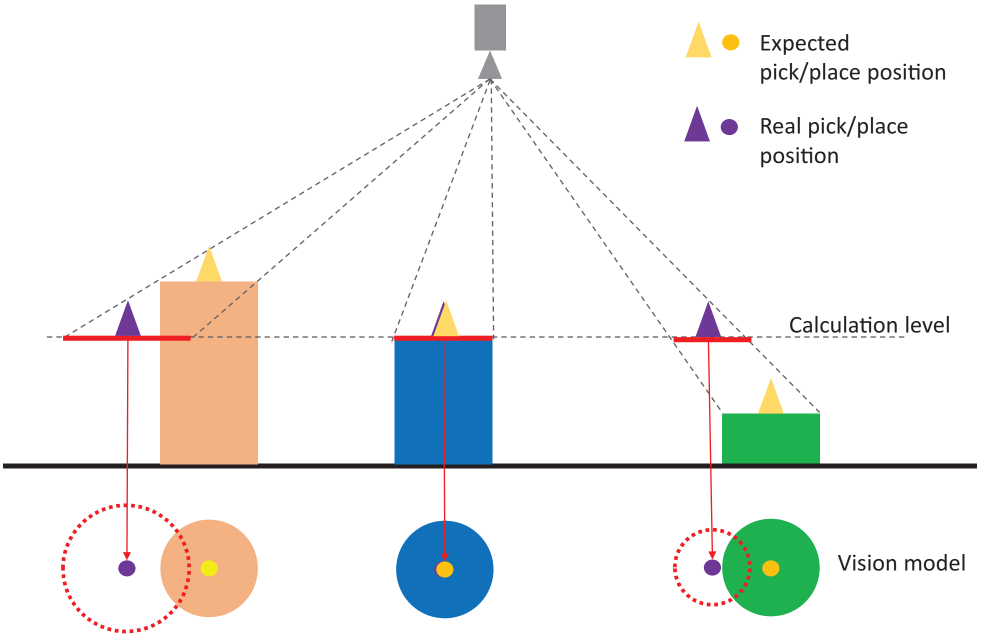
Introduction to height settings

The vision tools in PickMaster Recipe Manager typically return a result in a 2D coordinate system: X and Y angle, based on a calculation made at a certain height. The trained model is assumed to be located in the plane of the camera calibration.

Working with objects located above or below the calibrated plane will result in parallax position problems.

Assuming a calculation on a defined height, any object of a different height will be shifted by the resulting parallax.

The camera is taught to calculate based on the blue block top surface. Without the parallel, the camera will employ this surface for the lower green block or higher orange block. As a result, x- and y-coordinates for the green block and the orange block would move based on the misused calculation plane. That would mislead the robot to target a wrong position.



xx2300001744

With 2.5D calibration, a full 3-dimensional space is calibrated, which allows the system to compensate for the parallax error, given that the system knows the correct height of the object. This raises the following questions:

1. At what height is the object located (z-coordinate)?
2. What are the true x- and y-coordinates? The object recognition tools assumes that the object is still located in the calibration plane, and thus will provide coordinates projected on this plane.

To calculate the true x- and y-coordinates the camera's height above the calibration plane, and the product's distance (above/below) to the calibration plane must be

Continues on next page

known, based on the camera location, provided by performing a multi-view calibration. For more information, see *Calibrating the camera* in PickMaster® Twin - PowerPac Application manual.

Determining the height at which an object is located can be done in three ways with PickMaster Recipe Manager.

1. Manual input
2. Automatic calculation based on the scale change in relation to the trained object.
3. External input

All three methods will return the parallax compensated x- and y-coordinates, and method 2 and 3 will also return an estimated z-coordinate.

Effectively, the tools described in this section can be used to compensate for parallax error (find the true x- and y-coordinates) and for determining the height of a product.

Prerequisites

The camera must be calibrated with multiple images (Multi-view).

The height settings can be used together with a geometric standalone model, or main geometric-based inspection model.

Note

The main geometric-based inspection model does not compatible with Vision Height or External height.

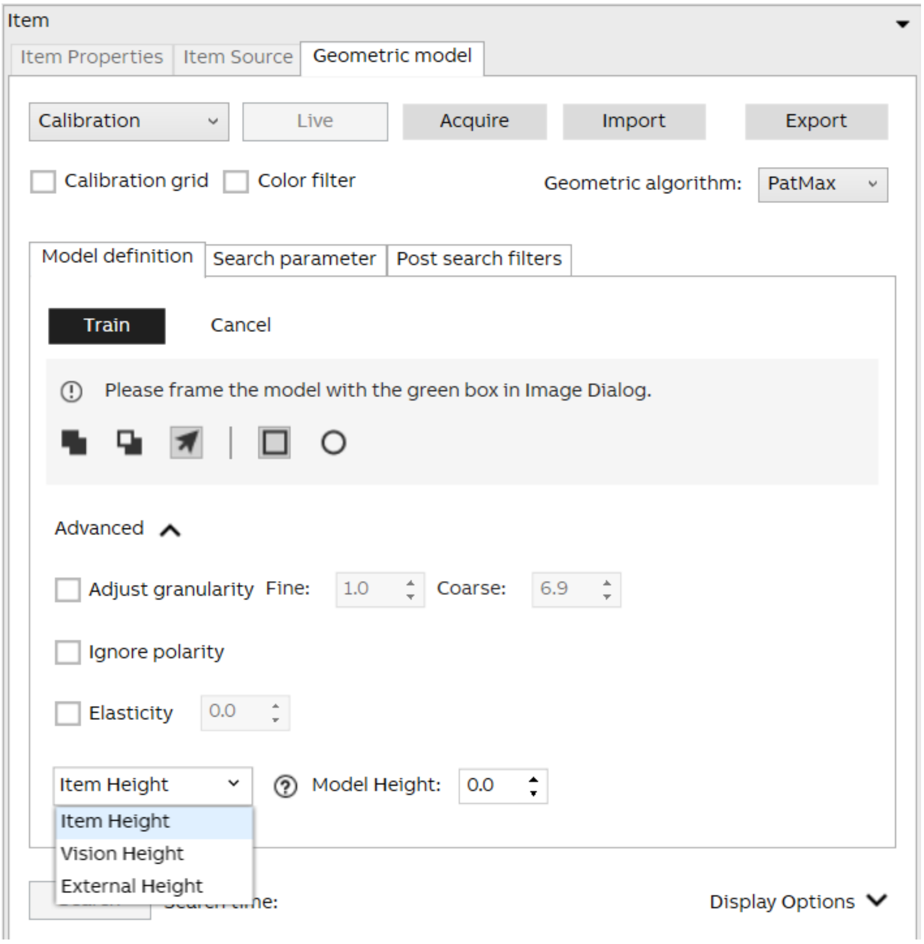
Configuring height settings

The height settings belong to a specific model and can only be configured together with **Geometric**.

Use this procedure to configure the height settings.

1. Create a **Geometric** model.
2. In the **Image** part, select calibration from the **Calibration** list. This must be a multi-view calibration.
3. In the **Model definition** part, click **Advanced**. This opens the **Geometric advanced model settings** dialog.
4. Choose an appropriate calculation method before training the item.

Continues on next page



Item

Item Properties | Item Source | Geometric model

Calibration | Live | Acquire | Import | Export

□ Calibration grid □ Color filter Geometric algorithm: PatMax

Model definition | Search parameter | Post search filters

Train Cancel

Please frame the model with the green box in Image Dialog.

Advanced

□ Adjust granularity Fine: 1.0 Coarse: 6.9

□ Ignore polarity

□ Elasticity 0.0

Item Height Model Height: 0.0

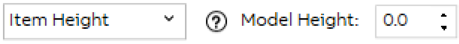
- Item Height
- Vision Height
- External Height

Display Options

xx2300001532

5 If the calculation method is set as **Item Height**,

Item height: Manually enter the value for the picking/placing height.

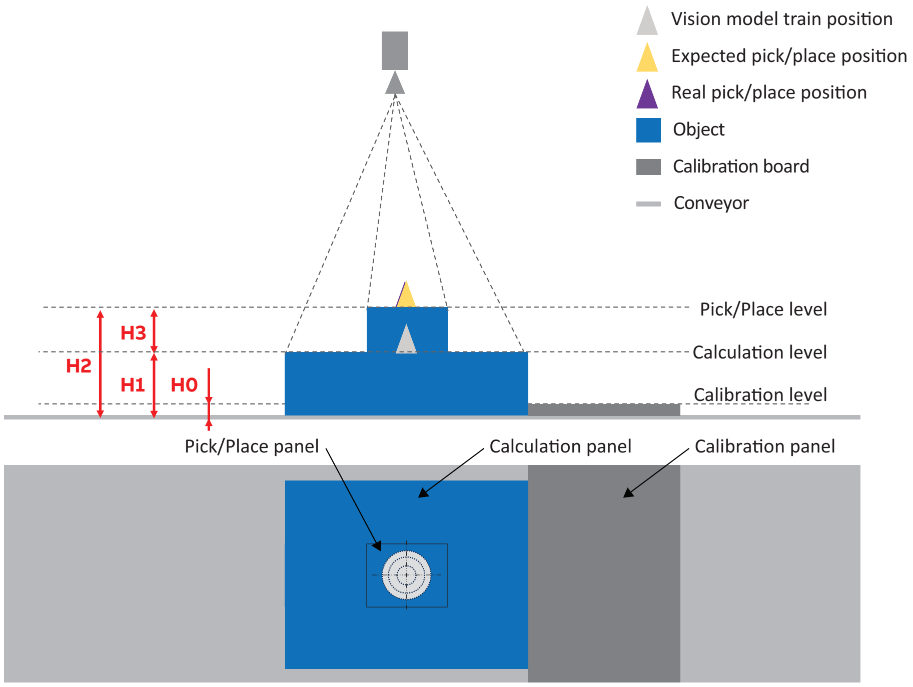


Item Height Model Height: 0.0

xx2300001533

One parameter should be fulfilled for this calculation method. **Model Height** is literally used to describe the height from the calibration panel to the calculation plane. Z-coordinate is defined as the true picking/placing height for the object and would be sent to the robot controller.

Continues on next page



xx2300001745

a Measure the height (H0) of the calibration board.

> **Tip**
>
> If the thickness of the calibration board is too thin to measure, for example a normal paper, then the user can ignore the height of calibration board and H0 is 0.

b Measure the height (H1) from the conveyor to the maximum contour panel for the picking/placing object.

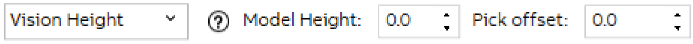
c Measure the height (H2) from the conveyor to the picking/placing panel for the picking/placing object.

d Enter the value (H1-H0) to the **Model Height**.

e Enter the value (H2-H0) to z-coordinate of the item setting **Size(x,y,z)[mm]/RH Size[mm]**. See *Adding an item on page 27*.

6 If the calculation method is set as **Vision Height**,

Vision height: The value from the calibration panel to the calculation plane is calculated from the scale change (relative to the trained pattern) of the found object.



Vision Height

Model Height: 0.0

Pick offset: 0.0

xx2300001534

Two parameters should be fulfilled for this calculation method. **Model Height** follows the same meaning defined in the **Item Height**. **Pick Offset** is the deviation from the calculation plane to the picking/placing panel. The

Continues on next page

calculation plane is defined as the maximum contour panel of the identified object.

> **Note**
>
> Enable uniform scale must be enabled. The maximum and minimum values must allow for sufficient scale variation.

a Measure the height (H0) of the calibration board.

b Measure the height (H1) from the conveyor to the maximum contour panel for the picking/placing object.

c Measure the height (H3) from the maximum contour panel to the picking/placing panel of the picking/placing object.

> **Tip**
>
> If the picking/placing panel is higher than the calculation panel on the z-direction, H3 is a positive number.
>
> If the calculation panel is higher than the picking/placing panel on the z-direction, H3 is a negative number.

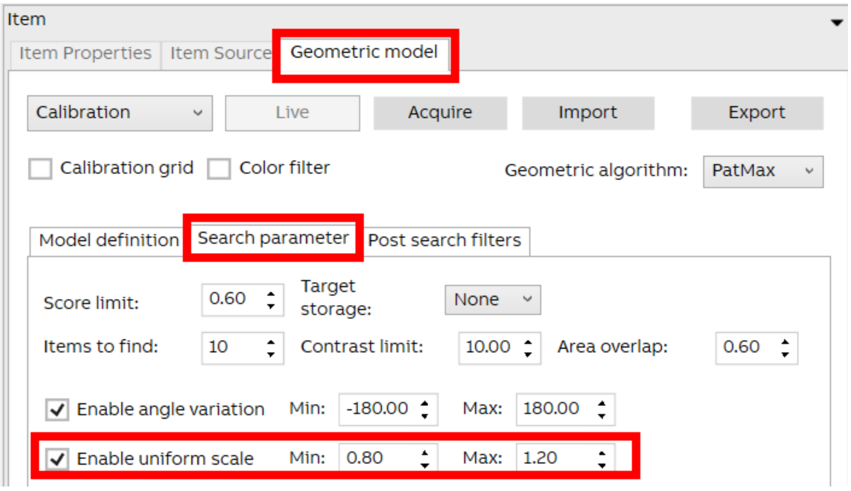
d Enter the value (H1-H0) to the **Model Height**.

> **Tip**
>
> If this value is a positive number, that means the calculation panel is higher than the calibration panel on the z-direction.
>
> If this value is a negative number, that means the calibration panel is higher than the calculation panel on the z-direction.

e Enter the value (H3) to the **Pick Offset**.

f Enable the **Enable Uniform Scale** and enter a proper range for the scaling.

Continues on next page



Calibration

□ Calibration grid □ Color filter

Geometric algorithm: PatMax

Score limit: 0.60

Target storage: None

Items to find: 10

Contrast limit: 10.00

Area overlap: 0.60

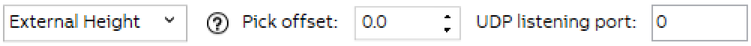
☑ Enable angle variation Min: -180.00 Max: 180.00

☑ Enable uniform scale Min: 0.80 Max: 1.20

xx2300001537

7 If the calculation method is set as **External height**,

External height: The product’s distance (above/below) to the calculation plane is calculated by the external source. This may be a height sensor or information from a cell PLC or any other external device. The z-coordinate is sent through a UDP port from external source to PickMaster Runtime. The UDP listening port for the external source should be unique for each vision model. Only the position message for current vision model can be sent through this vision model's listening port.



External Height

Pick offset: 0.0

UDP listening port: 0

xx2300001535

One parameter should be fulfilled for this calculation method.

a Enter the UDP port in **UDP Listening Port**. Then the calculated z-coordinate will be sent to PickMaster Runtime with the UDP message through this port.

Tip

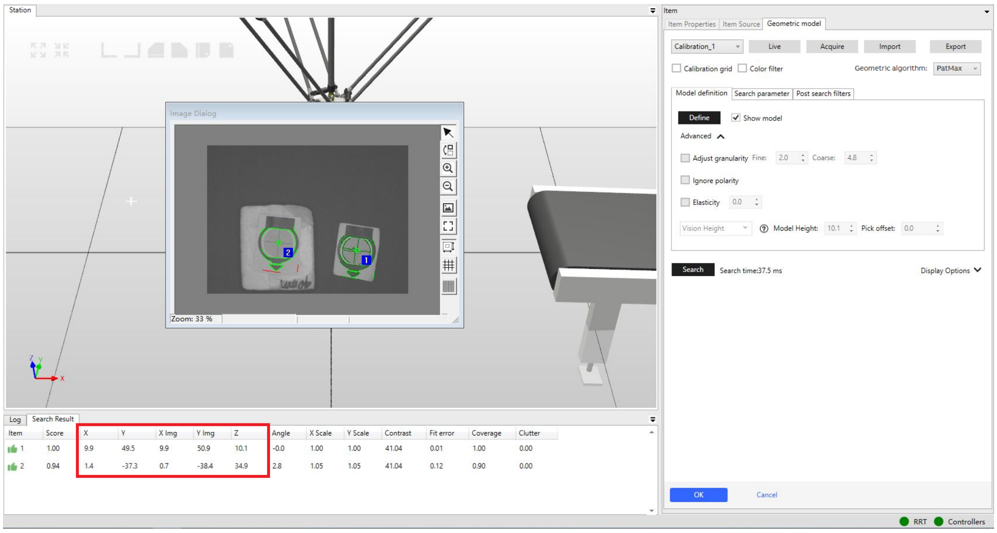
This **UDP Listening Port** should match with the predefined port in external source.

Note

Different vision models must be configured to use different ports.

With the height setting configured during the model training, the search results will contain the space information for all searched objects.

Continues on next page



□ Calibration grid □ Color filter Geometric algorithm: PatMax

☑ Show model

□ Adjust granularity Fine: 2.0 Coarse: 4.8

□ Ignore polarity

□ Elasticity 0.0

Vision Height Model Height: 10.1 Pick offset: 0.0

Search time:37.5 ms

Item	Score	X	Y	X Img	Y Img	Z	Angle	X Scale	Y Scale	Contrast	Fit error	Coverage	Clutter
1	1.00	9.9	49.5	9.9	50.9	10.1	-0.0	1.00	1.00	41.04	0.01	1.00	0.00
2	0.94	1.4	-37.3	0.7	-38.4	34.9	2.8	1.05	1.05	41.04	0.12	0.90	0.00

xx2300001536

Note

The Vision height method may be inaccurate. The accuracy depends on many factors such as camera the camera calibration, camera resolution, model size relative to image etc, thus the obtainable accuracy must be tested for a specific application.

Note

Defining a value for **Model height**, and selecting **Item height** as height method results in parallax compensation but no z-coordinate is calculated by the vision system.

Note

If there is only one object type, and it is always located at the same height, it is most accurate to calibrate the camera at this height instead of using **Model height** to compensate.

Tip

To filter out erroneous height information when using the **Vision height** method, set appropriate scale limits under the **Post search filters** part in the **Geometric model** dialog.

External height protocol

SetHeight

Request message sent to PickMaster Runtime to set the model height value from external equipment.

Message length: 5 bytes.

Continues on next page

Request example: Sending height 110.11 `[0x01, 0x42, 0xDC, 0x38, 0x52]`

Position	Parameter	Type	Variable attributes
Byte 0	`1`	Integer	The command to set the height. In the example, value `0x01` means the external source would set the height value.
Byte 1-4	`y,y,y,y`	Float	Item height in mm, big endian. In the example, byte value `0x42, 0xDC, 0x38, 0x52` indicates height value as 110.11

Response message received from PickMaster Runtime.

Message length: 5 bytes.

Response example: Runtime received the sent height 110.11 `[0x01, 0x42, 0xDC, 0x38, 0x52]`

Position	Parameter	Type	Variable attributes
Byte 0	`1`	Integer	The command to set the height. In the example, value `0x01` means the external source would set the height value.
Byte 1-4	`y,y,y,y`	Float	Item height in mm, big endian. In the example, byte value `0x42, 0xDC, 0x38, 0x52` indicates height value as 110.11

Tip

When converting from float type to byte stream, sort in large end order.

GetHeight

Request message sent to PickMaster Runtime to get the current height value set from external equipment.

Message length: 1 bytes.

Request example: Requiring height `[0x02]`

Position	Parameter	Type	Variable attributes
Byte 0	`2`	Integer	The command to get the height. In the example, value `0x02` means the external source would get the height value.

Response message sent from PickMaster Runtime.

Message length: 5 bytes.

Response example: Runtime sent the height 110.11 `[0x02, 0x42, 0xDC, 0x38, 0x52]`

Position	Parameter	Type	Variable attributes
Byte 0	`2`	Integer	The command to get the height. In the example, value `0x02` means the external source would get the height value.
Byte 1-4	`y,y,y,y`	Float	Item height in mm, big endian. In the example, byte value `0x42, 0xDC, 0x38, 0x52` indicates height value as 110.11

This page is intentionally left blank

Index

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

ABB AS
Robotics & Discrete Automation
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong New District
SHANGHAI 201315, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics & Discrete Automation
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics